NetOps and SecOps Guide to Public Cloud Journeys

How organizations adopt cloud: from hybrid migrations to cloud native deployments

Version 2.0
Released: February 23, 2021

Securosis, L.L.C. 515 E. Carefree Highway Suite #766 Phoenix, AZ 85085 **T** 602-412-3051 info@securosis.com www.securosis.com

Author's Note

The content in this report was developed independently of any sponsors. It is based on material originally posted on the Securosis blog, but has been enhanced, reviewed, and professionally edited.

Special thanks to Chris Pepper for editing and content support.

Licensed by Gigamon

Gigamon® delivers the industry's first elastic visibility and analytics fabric built to address the critical cloud visibility gap. We close this gap by enabling cloud tools to see the network and network tools to see the cloud. We do this through elastic scale-out and scale-up visibility and by delivering critical security and operational analytics across the hybrid cloud. By closing this gap, Gigamon has helped thousands of the world's leading organizations to run fast, stay secure and accelerate innovation in the digital era. For the full story on how Gigamon can help you, please visit www.gigamon.com.

Copyright

Table of Contents

Your Cloud Journey is Unique, but Not Unknown

Cloud computing is different, disruptive, and transformative. It has no patience for traditional practices or existing architectures. The cloud requires change, and there is a growing body of documentation on end states you should strive for, but a lack of guidance on how to *get there*. Cloud computing may be a journey, but it's one with many paths to what is all too often a highly nebulous destination.

Although every individual enterprise has different goals, needs, and capabilities for their cloud transition, our experience and research have identified a series of fairly consistent patterns. You can think of moving to cloud as climbing a mountain with a single peak, with everyone starting from the same trailhead. But this simplistic view, which all too often underlies conference presentations and tech articles, fails to capture the unique opportunities and challenges facing you. At the other extreme, we can think of the journey as involving a mountain range with innumerable peaks, starting points, and paths... and a distinct lack of accurate maps. This is the view which tends to produce hands thrown up in the air, expressions of impossibility, and analysis paralysis.

But our research and experience guide us between those extremes. Instead of a single constrained path which doesn't reflect individual needs, or totally individualized paths which require you to build everything and relearn every lesson from scratch, we see a smaller set of common options with consistent characteristics and experiences.These won't cover every option, but can be a surprisingly useful way to help structure your journey.

Introducing Cloud Adoption Patterns

Cloud adoption patterns represent a consolidated set of cloud adoption journeys, compiled through discussions with hundreds of enterprises and dozens of hands-on projects. Less concrete than specific cloud controls, they are a more general way of predicting and understanding the problems facing organizations when moving to cloud, based on starting point and destination. These patterns have different implications across functional teams, and are especially useful for network operations and network security, because they tend to fairly accurately predict many architectural implications, which then map directly to management processes.

For example there are huge differences between a brand-new startup or cloud project without any existing resources, a major data center migration, and a smaller migration of key applications. Each case migrates an existing application stack into the cloud, but the different scope and time constraints dramatically affect the actual migration process.

Using Cloud Adoption Patterns

The adoption patterns offer a framework for thinking about your upcoming (or in-process) journey, and help identify both strategies for success and potential failure points. These aren't proscriptive like the Cloud Security Maturity Model or the Cloud Controls Matrix — they won't tell you exactly which controls to implement, but are more helpful when choosing a path, defining priorities, mapping architectures, and adjusting processes.

These patterns represent a set of characteristics we consistently see based on how organizations move to cloud. Any individual organization might experience multiple patterns across different projects. For example a single project might behave more like a startup, even while you concurrently run a larger data center migration.

This research should help you better understand what approaches will work best for *your project in your organization*.

Cloud migrations can be tough and confusing. Nothing lets you skip over the hard work, but learning the lessons of those who have already climbed the mountain can save costs, reduce frustration, and increase your odds for success.

The Impact of COVID-19 on Cloud Adoption

The easiest way to describe the (cloud) impact of COVID-19 is to say that COVID forced many organizations to implement their 3-5 year digital transformation plans in 3-5 months. Or, in some cases, a few weeks.

The pandemic shut down offices, stripped data centers to skeleton crews, and redefined work from home. This forced digital transformation to accelerate in three ways:

- New cloud services, especially collaborate SaaS tools, were adopted to support a massive expansion in work from home employees.
- In parallel, organizations dramatically increased their self-hosted or cloud-based VPN capacity. This complicated network architectures and security controls.
- Timelines for IaaS cloud migrations significantly compressed. Two factors increased the pace of these migrations - greater reliance on remote IT workers who can more-easily manage cloud environments via the Internet/VPNs than on-site staff, and increasing motivation to reduce reliance on physical data centers (and the concordant staffing).

From a practical standpoint we saw a large increase in the number of lift and shift projects. To put it in the terms we define in the next section, planned data center transformation projects became closer to snap migrations — quick moves of existing resources into IaaS providers under tight timelines. To address this we have added more recommendations on network security and operations for lift and shift patterns.

The Four Cloud Adoption Patterns

Understanding Cloud Adoption Patterns

Cloud adoption patterns represent the most common ways organizations move from traditional operations into cloud computing. They contain hard lessons learned by those who went before. While every journey is distinct, hands-on projects and research have shown us a broad range of consistent experiences, which organizations can use to better manage their own projects. The patterns won't tell you exactly which architectures and controls to put in place, but they can serve as a great resource to point you in the right general direction and help guide decisions.

Characteristics of Cloud Adoption Patterns

We will get into more descriptive detail as we walk through each pattern, but we find this grid useful to define the key characteristics.

Characteristics	Developer Led	Data Center Transformation	Snap Migration	Native New Build
Size	Medium/Large	Large	Medium/Large	All (project-only for mid-large)
Vertical	All (except financial and government)	All	Including financial and government	Variable/All
Speed	Fast then slow	Slow (2-3 years or more)	18-24 months	Fast as DevOps
Risk	High	Low(er)	High	Variable
Security	Late	Early	Trailing	Mid to late
Network Ops	Late	Early	Early to mid	Late (developers manage)
Tooling	New, and old when forced	Culturally influenced; old & new	Panic (a lot of old)	New, unless culturally forced to old
Budget Owner	Project based/no one	IT/Ops/Sec	IT or poorly defined	Project-based; some security for shared services

- **Size:** The size of organizations likely to follow this pattern. For example developer-led projects are rarely seen in small startups because they can skip directly to native new builds, but common in large companies.
- **Vertical:** We see these patterns across all verticals, but in highly-regulated ones like financial services and government, certain patterns are less common due to tighter internal controls and compliance requirements.
- **Speed:** The overall velocity of the project, which often varies during the project lifetime. We'll jump into this more, but an example is developer-led, where initial setup and deployment are very fast, but then wrangling in central security and operational control can take years.
- **Risk:** This is a combination of danger to the organization and likelihood of project failure. For example in a snap migration everything tends to move faster than security and operations can keep up, which creates a high chance of configuration error.
- **Security:** When security is engaged and starts influencing the project.
- **Network Ops:** When network operations becomes engaged and starts influencing the project. Security folks are used to being late to the party because developers can build their own networks with a few API calls, but this is often a new and unpleasant experience for networking professionals.
- **Tooling:** The kind of tooling used to support the project. "New" means new, cloud-native tools. "Old" means the tools you already run in your data centers.
- **Budget Owner:** Someone has to pay at some point. This is important because it represents potential impact on your budget, and tends to indicate who has the most control over the project.

The Four Cloud Adoption Patterns

It's time to describe what the patterns look like and identify key risks. Our next section will offer some top-line recommendations to improve your chances of success.

Developer Led

Developer-led projects are when a developer or team builds something in a cloud on their own, and central IT is then forced to support it. We sometimes call this "developer tethering", because these often unsanctioned and/or uncoordinated projects anchor an organization to a cloud provider, dragging the rest of the organization in after them. These projects aren't always against policy — this pattern is also common in mergers and acquisitions.

This pattern drove a large portion of initial cloud adoption. In 2021 we see it less frequently as the driver to migrate to cloud, but it is growing as a source for inadvertently pulling organizations into multi-cloud deployments. Developers may jump outside the primary cloud to use a feature in a competing service, which then drags the entire organization across.

This creates a series of tough issues. To meet the definition of this pattern we assume you can't just shut the project down, but actually need to support it.

- **Size:** We mostly see this pattern in medium and large organizations. In smaller enterprises the overall scope of what's going on is easier to understand, whereas larger organizations tend to have an increasing number of teams operating at least semi-independently. Larger organizations are also more likely to engage in M&A activity which forces them to support new providers.
- **Vertical:** This pattern is everywhere, but less common in highly regulated and tightly governed organizations — particularly financial services and government.
- **Speed:** In the beginning, at least once security and networking find out about the project, there is a big rush to manage the largest risks and loop the project into some sort of central management. This flurry of activity then slows down into a longer, more methodical wrangling to bring everything up to standard. It starts with stopgaps, such as opening up firewalls to specific IP ranges or throwing in a VPC connection, followed by a longer process to rework both the deployment and internal support, such as by setting up hybrid direct connect.
- **Risk:** These are high-risk situations. Security was likely not involved, and we often find a high number of configuration errors when assessing these environments. They can often function as an isolated outpost for a while, but there are still risks of failed integration when the organization tries to pull them back into the fold.
- **Security:** Security is typically involved only late in development or after deployment, because the project team was off running on its own.
- **Network Ops:** As with security, networking enters late. If the project doesn't require connectivity back to an existing network they might not be involved at all.
- **Tooling:** Most often these projects leverage integrated tools provided by the cloud service provider. There is rarely budget for security or network specific tooling beyond that, since CSP tool costs are all hidden within basic cloud deployment costs. One problem we sometimes see is that after the project is discovered and there's the mad rush to bring it under central management, a bag of existing tools — which often fit the cloud platform poorly — are forced into place. This is most common with network and endpoint security tools which aren't cloud native — a virtual appliance isn't necessarily a good answer to a cloud problem.
- **Budget Owner:** The project team somehow managed to get budget to start the deployment, which they can use as a cudgel to limit external management. This may fall apart as the project grows and costs increase (as they always do) and the project has to steal budget from someplace else.

Key Risks

These should be obvious: you have an unsanctioned application stack running in an unapproved cloud, with which you may have little experience, uncoordinated with security or networking. However many project teams *try to do the right things*. You can't assume the project is an abject failure. Some of these projects are significantly better designed and managed, from the cloud standpoint, than lift and shifts or other cloud initiatives.

- Security configuration errors are highly likely.

- There may be unapproved and *ad hoc* network connections back to existing resources (hybrid cloud). At times these are unapproved VPN connections, SSH jump boxes, or similar.
- Deployment environments may be messy, full of cruft and design flaws.
- Development/testing and production are generally intermingled in the same account/subscription/project, which creates a larger blast radius for attacks.

Data Center Transformation

Data center transformations are long-term projects, where the migration is methodical and centrally planned. That isn't always beneficial — these projects are often hindered by overanalysis and huge requirements documents, which can result in high costs and slow timelines. They also tend to create their own particular set of design flaws. In particular, there is often a focus on building a perfect landing zone or "minimum viable cloud" which replicates the existing data center, rather than taking advantage of native capabilities of the cloud platform.

The pandemic has directly affected many of these projects in 2020-2021 as organizations accelerated their data center transformation timelines. In those situations the lines between this pattern and the *snap migration* pattern blur, resulting in a mix of benefits and risks in these cases.

- **Size:** You need a data center to transform, so this pattern shows up at very large, large, and sometimes mid-sized enterprises.
- **Vertical:** This pattern is common across all verticals which meet the size requirements. Five years ago we weren't seeing it with regulated industries, but cloud computing has long since passed that limitation.
- **Speed:** These projects tend to move at a snail's pace. There are a lot of planning cycles, and building baseline cloud infrastructure, before any production workloads are moved. In some cases we see progressive organizations breaking things into smaller projects rather than shoehorning everything into one (or a small number of) cloud environments, but this is uncommon. Multi-year projects are the norm, although more agile approaches are possible.
- **Risk:** The risk of a security failure is lower due to the slower pace and tighter controls, but there can be high risk of project failure, depending on approach. Large monolithic cloud environments are highly prone to failure within 18-24 months. Compartmentalized deployments (using multiple accounts, subscriptions, and projects) have a lower chance of major failure.
- **Security:** Security is engaged early. The risk is that the security team isn't familiar or experienced with cloud, and may attempt to push traditional techniques and tools which don't work well in cloud.
- **Network Ops:** Like security, networking is involved early. And as with security, the risk is lacking cloud domain knowledge for an effective and appropriate design.
- **Tooling:** Tooling depends on culture, silos, and politics. There is excellent opportunity to use cloud-native tooling, including existing tools with cloud-native capabilities. But we also see frequent reliance on existing tools and techniques which aren't well suited to cloud and end up causing problems.

- **Budget Owner:** These projects tend to have a central budget, so shared service teams such as operations, networking, and security may be able to draw on this budget or submit their own requests for additional project funding.

Key Risks

There are two major categories of risks, depending on the overall transformation approach:

- Large, monolithic projects where you set everything up in a small number of cloud environments and try to make them look like your existing data center. These 'monocloud' deployments are slow and prone to breaking horribly in 18-24 months. IAM boundaries and service limits are two of the largest obstacles. Agility is also often reduced, which even pushes some teams to avoid the cloud. Costs are also typically higher. Organizations tend to find themselves on this path if they don't have enough internal cloud knowledge and experience.
- Discreet, project-based deployment transformations leverage some shared services, but application stacks and business units have their own cloud accounts/subscriptions/projects (under the central organization/tenant). This cloud-native approach avoids many problems of monolithic deployment, but brings its own costs and complexities. The flexible nature of software defined networks in the cloud is a complex problem, especially when different projects need to talk to each other and back to non-cloud resources, which many enterprises never move to the cloud.
- Both models always involve hybrid connections for migrations. The largest risks are either placing too-strict controls over the hybrid network boundary, blocking effective migrations, or being overly-permissive and creating unintended complexity and security risks.

Snap Migration

Snap migrations are the worst of all worlds. Massive projects driven by hard deadlines, they are nearly always doomed to some level of failure. In our experiences the decision-makers behind these projects rarely understand the complexity of migrating to cloud, and are overly influenced by executive sales teams and consultants. Not that they are always doomed to complete failure, but the margins are thin and you will be navigating a tightrope of risks.

Snap migrations are still being driven by the pandemic. As we mentioned earlier, we have even seen some planned out data center transformations quickly become a snap migration thanks to COVID-19. The risks in those situations tends to depend on how far along planning was in the first place.

There is a subset of this pattern for more limited projects which don't encompass absolutely everything. For example imagine the same contract renewal drive, but for a subsidiary or acquisition rather than the entire organization. The smaller the scale the lower the risk.

- **Size:** Mid to large. You need to be big enough to have data centers, but not so big that you own the real estate they sit on. A defining characteristic is that these projects are often driven by contract renewals on hosted or managed data centers. That's why there's a hard deadline...

management wants out, as much as possible, before they get locked into the next 7-year renewal.

- **Vertical:** Organizations across all verticals find themselves in hosting contracts they want out of. We even know of projects in highly regulated financial services which you'd think would never accept this level of risk. Government is the least likely, and tends to be driven more by whichever political appointee decides they want to shake things up.
- **Speed:** 18-24 months for the first phase. We rarely see less than 12 months. Sometimes there will be a shorter initial push to get out of at least some data centers, as the contract moves into a month-by-month extension phase.
- **Risk:** As high as it gets in every possible way. Organizations falling into this pattern might have some internal cloud experience, but as a rule not enough people with enough depth to support the needed scale. There is heavy reliance on outside help, but few consulting firms (or cloud providers themselves) have a deep bench of solid experts who can avoid all the common pitfalls.
- **Security:** Security is somewhat engaged but can't do anything to slow things down. They are also typically tasked with building out their own shared services, so likely aren't staffed to evaluate individual projects. They tend to trail behind deployments, trying to assess and clean things up after the fact. They often get to set a few policies up front but until they stand up their own monitoring and enforcement capabilities, things slip through the cracks.
- **Network Ops:** There is a bit more variability here, depending on the deployment style. If there is a monocloud (or small number of environments), networking is typically engaged early and plays a very strong role in getting things set up. They are tasked with configuring the fatter pipes needed for such large migrations. The risk is that they often lack cloud experience, and introduce designs which work well in a data center but fit cloud deployments poorly.
- **Tooling:** Panic is the name of the game. The initial focus is on the tools at hand, and vendors already in place, combined with cloud-native tools. We hate to say it, but this can be deeply influenced not only by culture but by which consultants are already in the door. Eventually the project starts introducing more cloud-native tooling to solve specific problems. For example in projects we've seen, visibility (cloud assessment and mapping) tools tend to be early buys.
- **Budget Owner:** This can be poorly defined, but they often pull from a central IT budget or specially designated project budget. Whoever controls the money has the most influence. The chances of success go up when all teams are properly funded and staffed. Also, water is wet.

Key Risks

They abound but we can categorize them based on project characteristics:

- As any IT pro knows, every project of this scale runs over time and budget.
- There is often a reliance on outside contractors who push things along quickly, but don't know (or care) enough to have a sense of the enterprise risk. Their job is to get things moved — not necessarily to do so the safest way. This can lead to exposure as they accept risks a company employee might avoid.
- Security often lacks general cloud security knowledge, as well as provider and platform experience. They can build this but it takes time, and in the process the organization is likely to

accumulate technical security debt. For example two of the most common flaws we find on assessments are overly-privileged IAM and poorly segregated networks.

- Rapidly designing a cloud network at scale is difficult and complex, especially for a team which is still keeping the existing environment running, and (like security) probably lacks deep cloud experience. We often see one or two members of a team tasked as cloud experts but this really isn't enough. Given the time constraints the network often ends up poorly compartmentalized, and projects tend to be shoveled into shared VPCs/VNets in ways which later run up against service limits, performance problems, and other constraints.
- Hybrid connections are more likely to be sloppy and wide open, due to the demands to move things quickly. This may result in security exposures due to lack of appropriate internal network controls on both the cloud side and the hybrid network boundary between the environments.

Native New Build

Native new build projects are true cloud-native deployments. That doesn't mean they are all brand new projects — this pattern also includes refactoring and rearchitecting, so long as the eventual design is cloud native. These may also include hybrid deployments — the new build may still need connections back to the premises.

- **Size:** All sizes. In a large enterprise this will likely be a designated project or series of projects (especially in a "new to cloud" organization). In a small startup the entire company could be a new build.
- **Vertical:** All verticals. Even government and highly regulated industries. We have worked on these projects with financials, state governments, and even public utilities.
- **Speed:** As "fast as DevOps". We don't mean that facetiously — some teams are faster and some slower, but we nearly always see DevOps techniques used, and they often define the overall project pace. *These are developer-driven projects*.
- **Risk:** We will talk more about risk in a moment, but here we'll just note that risk is highly variable, dependent on the skills and training of the project team.
- **Security:** Unlike our previous example security may be late to the project. There is usually an inflection point, when the project is getting close to production, at which security gets pulled in. Before that the developers themselves manage most security. This improves over time — the organization is more likely to struggle in this area on early projects, but start integrating security earlier over time, as more and more moves to cloud and skills and staffing improve.
- **Network Ops:** Networking is more likely to be engaged early if there are hybrid connectivity requirements, or might not be involved at all depending on the overall architecture. These days we see a growing number of serverless (or mostly serverless) deployments where there isn't even a project network, and all the components talk to each other within the "metastructure" of the cloud management plane.
- **Tooling:** Typically newer, cloud native, and often CSP provided. Quite a few of these projects start in their own cloud silos and use the provider's tooling, but as more of these projects deploy there is increased demand for central management and shared services (such as security assessment) to be added. We sometimes see development teams forced to use traditional on-

premise tools, but this tends to be cultural — it isn't usually the best solution to the problems at hand.

- **Budget Owner:** Project based. Once you do enough of these there will also be shared services budgets for teams like security and networking.

Key Risks

Despite our optimistic opening, these projects bring their own risks. The project may be well-segregated in its deployment environment (an Azure subscription in our example) but that doesn't mean developers won't be over-provisioned. Or that a PaaS endpoint in the VNet won't ignore the expected Network Security Group rules (yes, that happens).

- This pattern can carry all the risks of the developer-led pattern if it is poorly governed. We have seen large organizations running dozens or hundreds of these projects, all poorly governed, each carrying tons of risks. If you read about a big cloud breach at a client who was proudly on stage at their cloud provider's conferences, odds are they are poorly governed internally.
- Cloud native services have different risks which take time to understand and learn to manage. In the data center migration pattern there is less reliance on the latest and greatest "serverless this, AI that", so traditional security and management techniques can be more effective. With native new builds you may be using services the cloud provider itself barely understands.
- Friction between security and the project team can badly impact the final product. Overly proscriptive security pushes teams into workarounds. Early partnering, ideally during initial development of the architecture, with security pros trained on the cloud platform, reduces risk.
- Managing a lot of these projects at scale **is really really hard**. Setting up effective shared services and security and network support (especially when hybrid or peered networks are involved) take deep expertise. Cloud providers are often terrible at helping you plan for this — they just want you to move as many workloads to them as quickly as possible.

Integrating Cloud Implementation Patterns

Cloud adoption patterns describe the big picture ways organizations move into cloud computing. Cloud implementation patterns form the next layer down and describe the implementation of individual applications, projects, and workloads. Adoption patterns are used by entire organizations, while implementation patterns are used on individual projects. There are many different implementation patterns, but we tend to lump them into 2 broad categories: *cloud native (*also called *native new build,* just like our adoption pattern*)*, or *lift and shift*. Then, within lift and shift, we focus on three primary options: *rehost, refactor, and rebuild*.

Cloud Native vs. Lift and Shift

When building with Infrastructure and Platform as a Service one of your first decisions is to decide if you want to base it off an existing, usually traditionally-hosted design, or to design something completely new and native for your selected cloud service provider. This decision is made in every one of our adoption patterns

If you're starting from scratch the decision is pretty simple and you are always creating a new build, but you still have to choose *how* cloud native the design will be. For example, nothing prevents you from designing and building something that looks exactly as it might in a data center. We see less and less of this over time since this approach rarely makes any sense from a financial, performance, or overall benefits standpoint. Most cloud native designs take all or some advantage of the CSPs services like serverless, software defined networks, database platforms, and some of the hundred or more PaaS offerings every major provider includes.

The native new build approach is great for brand new applications, but the majority of cloud deployments today are application stacks originally created for a data center that are then migrated to cloud. We refer to these as *lift and shift* since we are lifting the running stack and shifting it to the cloud, just as we might migrate it to a new data center.

Lift and Shift Patterns

While there are a lot of lift and shift options, they can be summarized into three major patterns:

Rehost

We call this "Lift and Pray". Rehosting is when you take your existing application stack, including the underlying infrastructure, and replicate it completely in the cloud. In some organizations this entails

replicating their entire data center's network and infrastructure design and then moving everything over in virtual machines.

To be clear, rehosting is a profoundly poor decision in every single situation. Cloud platforms are not the same as physical data centers, and the low-level technical differences will have profound consequences. Organizations will commonly experience the following issues within 6-18 months of a rehosting project, with the problems appearing more quickly when the project is of a larger scale.

- Costs increase due to an inability to optimize for the cloud platform. Virtual machines are larger, network traffic is higher, and, in general, you are unable to use most of the cost-management capabilities of the platform since applications aren't designed for elasticity.
- Reliability decreases since assets are moved into a virtualized infrastructure. Rehosted applications are unable to use cloud native resiliency capabilities designed to compensate for this.
- Security tends to decrease due to the addition of the management plane and IAM requirements, but a continued reliance on traditional security tools. Also, quite a bit of cloud's security benefits come from leveraging the security capabilities and inherent hardness of PaaS.

One strategy to limit the negative impacts of rehosting is to map out the application stack and dependencies and then build *only* the minimum required infrastructure in an isolated cloud account (or network), then use peering, API gateways, or other techniques to connect the rehosted components back into other applications and services.

Refactor

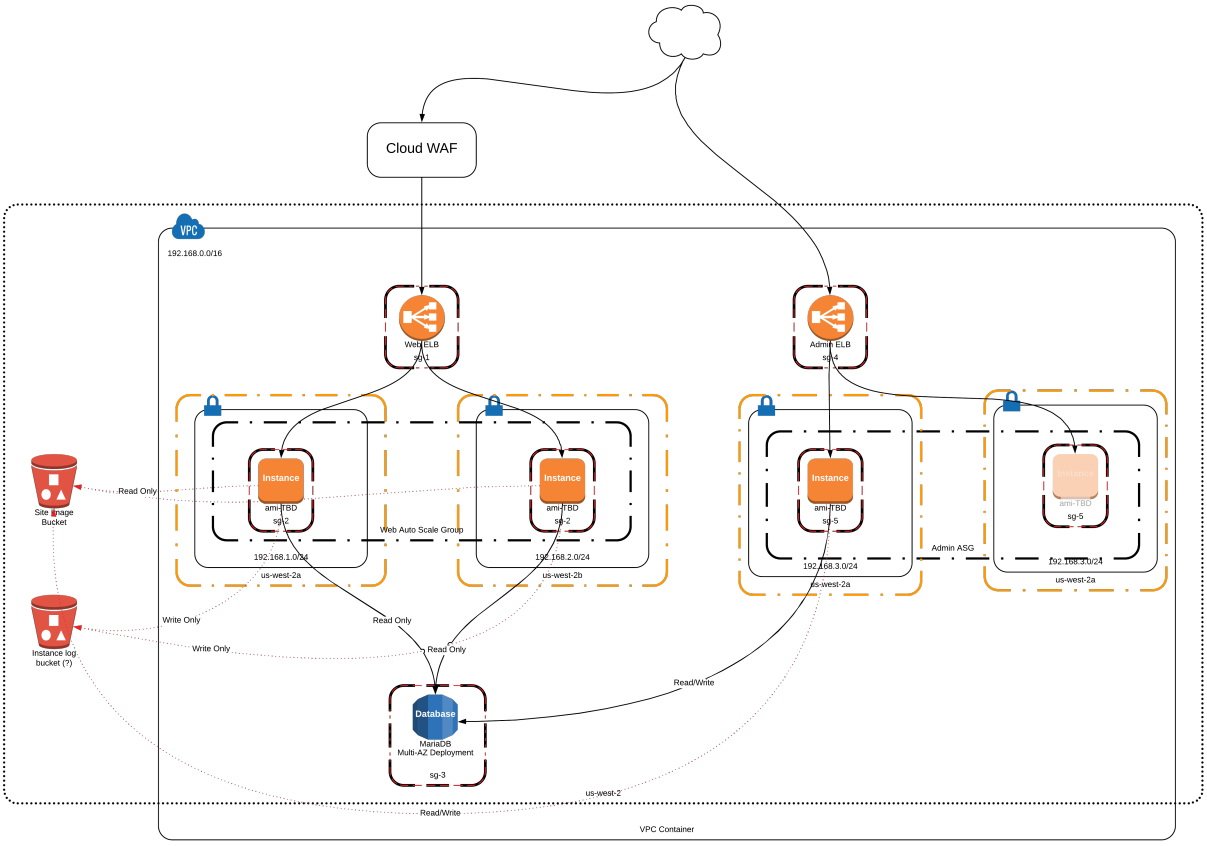
Refactoring is often the sweet spot for lift and shift migrations. In a refactor you start with a traditionally-hosted application, map it out, and then mix and match the existing resources and services with cloud-native options.

Some simple examples include:

- Building the application in a simplified and isolated virtual network that contains only the subnets, routes, and connections required for the application. Then locking down every layer with security groups for internal hardening that is cost-prohibitive in a physical network.
- Migrating load balancers and databases to the cloud-managed PaaS version of the same platform, increasing reliability, scalability, and security.
- Modifying application servers to run in an auto scaling groups to increase resiliency and elasticity. Security also increases if you use immutable concepts, like disabling SSH access.

This is the initial architecture used to migrate securosis.com from a high-security traditional data center into Amazon Web Services and demonstrates all of these refactors. Security was maintained, management costs decreased, and there was over a 10X reduction in hosting fees.

The key to refactoring is to treat it as an iterative process, constantly leveraging cloud native principles when they fit. Early quick wins tend to include moving into least-privilege, minimally-viable



networks, swapping in cloud load balancers, and moving to PaaS databases. None of these involve any serious application changes, but increase security and reliability.

Rearchitect

This is the lift and shift version of a native new build. An application is ripped apart and rebuilt as a pure, cloud native design. Imagine taking your server-heavy big data analytics platform and migrating it to a completely serverless architecture using various PaaS components from your cloud provider.

Organizations will often choose to rearchitect legacy applications that have outlived their effective lifespan. The old application is kept running in its existing environment while the new, rearchitected version is created in the cloud. We also see rearchitected projects run in parallel with both rehosted and refactored applications. The older application version is migrated into the cloud to meet timeline requirements, but then the new application stack is built in parallel and customers are later migrated over.

A true rearchitecture project starts from scratch, but we have personally been involved in iterative refactor projects that start as a rehost, and over time components are swapped in and out until the rearchitected end state is achieved.

Building Network Manageability and Security

Recommendations for a Safe and Smooth Journey

Learning cloud adoption patterns doesn't just help us identify key problems and risks — we can use them to guide operational decisions to address the issues they consistently raise. This research focuses on managing networks and network security, but the patterns include broad security and operational implications which cover all facets of your cloud journey. Governance issues aside, we find that networking is typically one of the first areas of focus for organizations, so it's a good target for our first focused research. (For the curious, IAM and compliance are two other top areas organizations focus on, and struggle with, early in the process).

The Power of the Minimum Viable Network

We recommend a concept we call the "Minimum Viable Network," for cloud architectures, reflecting that software defined networks (SDNs) are not subject to the same constraints as a physical network. This enables building custom networks with just the pieces needed to get the job done, and nothing more. SDNs also behave differently than physical networks and don't suffer from typical physical network security issues, such as ARP cache poisoning or even the ability to sniff network traffic.

The idea of a Minimum Viable Network is to determine the application architecture and then create *ONLY* the networking required to support that application, enforcing least privilege routing and security groups, and also utilizing PaaS components like cloud load balancers. *This effectively turns the packet switched network into a circuit switched network* in that application components can ONLY communicate to the other allowed components, and nothing can sniff the traffic in between (except, in some cases, the cloud provider). The network drops all other traffic. This eliminates the need for constructs like the traditional DMZ or network zones since the entire network itself is least privilege and default deny.

For example, you can create a traditional 3-tier application stack with a public facing Application Load Balancer, backed by a web server in a private subnet that ONLY accepts traffic from the load balancer. The web servers can only connect to application servers that accept connections from them. Similarly, the application servers can only send traffic to databases that allow such connections. Each tier is default deny and only allows inbound connections from the North and outbound connections to the South. In many cases you can even implement this without any

Internet connectivity (in private subnets) aside from the public load balancers, which are highly secure PaaS constructs maintained by the cloud provider.

Rather than building the network and putting applications inside, you design the applications and then fit the network to the application's needs.

This is the principle used in the securosis.com architectural diagram shown in the previous section. If we look at that design with a network security eye we see:

- Access is restricted from the cloud WAF so only clean traffic goes to the VPC.
- The application load balancers (ALBs) are the only resources in public facing subnets, and only allow 80/443 traffic.
- All the instances are in private subnets and only accept traffic from the ALBs.
- The "admin" instance accepts logins but only from a VPN hosted in a different environment.
- Not shown are the subnets for the RDS database, which are also private only, and only accept traffic from the instances. If any direct logins or RDBMS access for data massaging are needed the security group rules are changed for temporary access.
- Connections to S3 are made using a service endpoint, eliminating the need for a NAT Gateway for Internet access.
- SSH is disabled on the non-admin instances. These are autoscaled and immutable and are only modified by changing the base image.
- There are no self-referencing security groups (security groups allowing internal access) to prevent horizontal attacks. Security groups in AWS work on a resource level, not a subnet level, which is very powerful to block East/West attacks.
- This architecture dramatically reduces the overall attack surface. The network only allows the minimum required connectivity, and only contains the minimum required subnets and route tables to allow access. We have essentially fit the network to the application. In fact, the network was only designed after the application stack was architected.
- This is a very simple example of a small site, but the principles can apply to much larger architecture with a greater number of tiers, internal load balancers, API Gateways, and PaaS components.

Developer Led Recommendations

Having your entire organization dragged into the cloud thanks to the efforts of a single team is disconcerting, but not unmanageable. The following steps will help you both wrangle errant projects under control, and also build a base for moving forward:

- Remember that to fit this pattern you should be new to either the cloud in general, or to this cloud platform specifically. These are not recommendations for unsanctioned projects covered by your existing experience and footprint.
- Don't be antagonistic. Yes, the team probably knew better and shouldn't have done it... but your goal now is corrective action, not punitive.

- You need to reduce urgent risks while developing a plan to bring the errant project into the fold.
- Don't simply apply existing policies and tooling from other environments to this one. You need tooling and processes appropriate for this cloud provider.
- In our experience, despite the initial angst, these projects are excellent opportunities to learn your initial lessons on this platform, and to start building out for a larger supported program.

The following recommendations go a long way toward reducing risks and increasing your chance of success:

- Step one is to figure out what you are dealing with.
  - How many environments are involved? How many accounts, subscriptions, or projects?
  - How are the environments structured? This requires mapping out the application, the provider's PaaS services (such as load balancers and serverless capabilities), IAM, network(s), and data storage.
    - Pay particular attention to any hybrid connectivity, and if those connections comply with network operations and security policies.
  - How are the services configured?
  - How are the networks structured and connected? The Software Defined Networks (SDN) used by all major cloud platforms only look the same on the surface — under the hood they are quite a bit different.
  - And, most importantly, *where does this project touch other enterprise resources and data?!?* This is essential for understanding exposure. Are there unknown VPN connections?
- Then prioritize your largest risks.
  - Internet exposures are common and one of the first things to lock down. We commonly see resources such as administrative servers and jump boxes exposed to the Internet, plus PaaS services (we cover those in our last point). In nearly every single assessment we find at least one instance or container with port 22 exposed to the world. The quick fix for these is to lock them down to your known IP address ranges with security groups.
  - Identity and Access Management is the next big piece to focus on. This research is focused more on networking, so we won't spend much time on these here. But when developers build out environments they almost always over-privilege access to themselves and application components. Sweep out static credentials, enable federation, and turn on MFA everywhere you can.
  - Misconfigurations of cloud services, some of which become public on the Internet, are next. Public storage buckets, unsecured API gateways, and other services which are Internet exposed but *won't show up if you only look at the virtual networks*.
- After cleaning those up it's time to start layering in longer-term remediations and your gameplan. This is a huge topic so we will focus on network management and security.

- During early discovery of developer-led projects, it is very common to want to tie the errant cloud account back into your on-premise network for connectivity and management. This instinct is usually wrong. Networking wasn't involved at the start, so it is unlikely there is an established network connection, and adding one won't necessarily provide any benefit. While outside the scope of this research, a wide range of techniques is available to provide necessary services to disconnected cloud accounts... or cloud-native connections such as service endpoints which achieve the same goals without the heavy lifting on fat pipes and CIDR segmenting.
  - We aren't suggesting you don't manage the network — we are saying you don't need to simply wire it up to your existing infrastructure to manage it or the resources it contains.
  - A big complication for integrating an unplanned SDN is the existing IP addressing (if there's even a virtual network — a real question thanks to new serverless architectures). This may be further motivation to keep it as a separate enclave.
- Now it's important to fully map out all the internal connections, including connections between different virtual networks and accounts which are peered or otherwise connected using cloud-native techniques such as service endpoints.
  - One of the most common networking mistakes in this kind of project is too-open internal networks. Clouds default to least privilege, but it is still all too easy to just open everything up to reduce friction during development. Use your map to start compartmentalizing internally. This may include network structure changes (routing and subnet modifications).
  - Security groups should reference each other (in Azure you need Application Security Groups) instead of relying on IP addressing for internal cloud connections. This is fundamental to cloud networks, but not where people with traditional network security backgrounds tend to start.
- Virtual security appliances (we are mostly talking about firewalls, IDS, and IPS) should only be used when security groups and native cloud capabilities cannot meet your needs. Virtual appliances are expensive to run because cloud providers charge for their compute cycles, and they create unnecessary chokepoints which affect performance and reliability.
- Lastly, once everything is in a known good state, you should implement continuous configuration assessments and guardrails to keep things that way. For example in a production application you should generate an alert on any security group change, creation of new internet gateways, and other structural changes.

Overall the key to handling this situation is to avoid panic, focus on obvious risks first, and then take your time to sweep through the rest in as cloud and provider specific a way as possible. Use it as a base to build your program, understanding that you will need to make short-term sacrifices to handle any significant exposures.

Data Center Transformation Recommendations

Large multi-year data center moves are some of the most complex projects in information technology. Moving everything from one physical location to another is a massive undertaking. Doing so while keeping services up and running, without shutting the business down (either planned or unplanned), even more so. Swapping to an entirely different technology foundation at the same time? That can be the definition of insanity, yet every single organization of any size does it at some point.

The most common mistakes we see involve shoehorning traditional architectural and security concepts into the cloud — which can lead to extended timelines, increased costs, and long-term management issues. A few key principles can keep you moving in the right direction:

- If you are bad at network management and security in your existing data center, you will be surprised at how little changes in cloud. Look at cloud as an opportunity to do things better, ideally in a cloud-native way. Don't just bring across your existing practices without change — especially bad habits.
- Time is your friend. Don't rush, and don't let your cloud provider push you into moving faster than you are comfortable with. Their priorities are not yours.
- Don't assume your existing tools and processes will work well in cloud. Many organizations bring things across due to employee familiarity or because they already have licenses. Those aren't great reasons to deploy something in an entirely new operating environment.
  - That said, these days many products offer extensions for the cloud. You should still evaluate them instead of *assuming* they will meet your needs, but they might be a useful bridge.
- Learn first, move second. Take the time (if you have it) to hire and build the skills needed to operate on your new platform. You absolutely cannot expect your existing team to handle both the current environment and cloud if you don't give them the time to learn the skills and do the job.

In the *developer led* pattern we had to balance closing immediate risks against simultaneously building support for an entirely new operating environment and preparing for long-term support. Scary and difficult, but also usually self-constrained to something manageable like a single application stack. In a data center transformation the challenges are **scaling,** transitioning completely to a new environment, and any need to carry over legacy resources not designed to run in cloud.

- Start by building your plan.
  - You **will not** want to run everything in a single huge account/subscription/project on just 1-3 virtual networks. This is all too common and falls apart within 18-24 months due to service limits, differences in how cloud networks work, and cloud-native application requirements.
  - You will want multiple cloud environments (accounts/subscriptions/projects are the terms used by different providers) and very likely multiple virtual networks in each environment.

These are needed for blast radius control, managing service limits, and limiting IAM scopes.

- Map out your existing applications and environments (networks, cross-app connectivity, associated security controls, and related supporting services such as DNS and logging). Create a registry and then prioritize and sequence your moves.
- Map out your application dependencies. You might have 50 applications which all connect back to a shared customer database. This directly impacts how you structure your accounts, virtual networks, and connectivity options.
- Design a flexible architecture. Think of it as a scaffold to build on as you pull project by project into the cloud. You don't need to definitively plan out every piece of the migration before you move, unless you really like spending massive amounts of money on project managers and consultants.

- Then start building your scaffold.
  - Start with foundational shared services you will need across all your cloud environments: logging/monitoring receivers, cloud assessment (cloud security posture management), cloud automation (including cloud detection and response), other visibility/monitoring tools, and IAM.
  - Your will likely need at least one transit network (a central virtual network used to peer your other virtual networks, even across cloud environments). Design this network (in its own account) for transit only — not to contain any actual resources (except possibly some shared services).
  - Many shared services work better as "endpoint services", which are published within the cloud provider but don't require network peering outside. Implementation is quite different at each cloud provider, so we can't get more specific in this research, but endpoint services really enable you to take advantage of cloud software defined networks, and reduce reliance on fixed IP addresses and traditional network segmentation.
  - Build infrastructure as code templates for "landing zones" for the new accounts you will create for various projects. These can and should embed foundational security controls, such as links to transit networks and endpoint services (as appropriate), baseline network security controls, and implementation of the assessment, monitoring/logging, visibility, automation, IAM, and other core tools you use to track each of your environments.
  - Don't forget, these are just pointers to get you started — we aren't trying to downplay the complexity of these projects.
- With the scaffold in place, it's time to start migrating workloads.
  - This is an iterative process. Just as you build a scaffold and smaller environments, move your projects over in prioritized order to learn as you go.
  - As you move each project over, try to refactor and rearchitect to the best of your ability. For example you should "fit the network to the application" — you can now have multiple software designed networks, each containing the bare minimum to support one project. This really helps reduce attack surface and compartmentalize.

- Keep up with continuous assurance. Mistakes happen and your shared monitoring, visibility, and remediation tools will help reduce exposure. Don't wait until the end for one big assessment.

These migrations and transformations can be overwhelming if you try to plan everything out as one giant project. If you think in terms of building central services and a scaffold, then migrating projects iteratively, you reduce risk while increasing your chance of success.

Snap Migration Recommendations

Snap migration can be the riskiest of all adoption patterns. Short timelines, critical resources, and rarely the skills and staff needed. They combine the messiness of the developer-led pattern with the scale of data center transformation. In our experience these projects often include a heavy dose of cloud provider or consultant pressure to move fast and gloss over complexity.

Let's start with our principles:

- Your primary objective is to minimize immediate risk while creating a baseline to use as you clean things up over time after the cutover.
  - Get the right people with the right skills. This includes training, hiring, and consulting. Make sure you really vet the people you are bringing in — even your cloud provider's experts may be fresh out of school with little real-world experience.
  - Don't just copy and paste your existing network into the cloud. This approach *always* fails within 18-24 months, for many already-cited reasons.
  - Constantly look for opportunities to control blast radius. Use multiple virtual networks and accounts, and only connect them where needed.
  - You typically won't have time in a snap migration for any serious refactoring or rearchitecting. Instead focus on a strong scaffold and management controls, with the expectation that you can start making things a little more cloud native once the main cutover is complete.

These are simply bad situations, which you need to manage as well as possible. Some smart decisions early on will go a long way to helping you set yourself up for iterative cleanup after the mad rush is over.

- Start by building a scaffold — not a parking lot.
  - Follow our recommendations for the *data center transformation* pattern.
  - While you might need to replicate your current network, *nothing says you have to do that in a single virtual network*. With peering and transit networks, you can architect your new cloud network with subnets in separate virtual networks and accounts based on projects, then connect them together with your cloud provider's peering capabilities. For example you can create the 10.0.1.0/24 subnet in one virtual network in one cloud account, and the 10.0.2.0/24 subnet in an entirely different virtual network and account, then peer them together.

    - This improves your long-term security because account segregation, even across peered networks, helps manage the service limit and IAM issues which cause so many problems when everything is in one account. For example if different projects share the same virtual network, it is hard to designate IAM privileges so the various administrators cannot affect each other's resources.
    - Knowing your subnets and connectivity requirements are key factors for success.
- As with our data center transformation pattern, build your shared services after (or concurrently with) your network scaffold.
- Be cautious and judicious about allowing Internet access. Controlling the public perimeter early is crucial. Quite a bit can be accidentally opened up during data migration, as teams rush to throw assets into the cloud, so make sure you keep a continuous eye on things.
    - Also track network connections to your on-premise environments. At some point many of these openings should be shut down, as projects complete migration and no longer need to call back to the doomed data center.
- To the best of your ability, also implement in-cloud network segregation with security groups. Another issue we often see is excessive security group openings within the network — ops, devs, or even security may not know all the right port and protocol combinations for a given application. There is literally zero cost to more security groups, which are effectively firewalls around every resource. Use them to your advantage and dial down permissions.
- In the long term you will want to sweep through and refactor and rearchitect where you can. This is much easier if you migrated into multiple accounts and virtual networks.

Native New Build Recommendations

Cloud native doesn't mean a project is inherently secure, but it does completely shift the security and networking focus. The key principles are:

- *Cloud security and operations start with architecture and end with automation.* A well-designed architecture will reduce most risks. Automation maintains a strong and safe posture over time.
- Serverless, containers, and other emerging technologies are the norm. You may or may not have networks, but any networks you do have will be quite different from traditional infrastructure.
- Your public-facing perimeter is more than just what your virtual networks expose. Many services in cloud providers are (potentially) directly public-facing, so must be managed at the configuration level.
    - Subdomain takeovers in cloud are very common due to these services. Make sure you are monitoring at the DNS level — not just IP addresses.
- The biggest issues we see for this pattern are mostly related to governance. Dev teams are allowed to move fast and break things, and while there is nothing inherently wrong with that, it becomes a problem when they move faster than security can contain risk. Early

engagement, architectural support, continuous monitoring, and strong team relations are essential for success.

- Fit networks to applications. This is a core philosophy: start with the application's needs and build the network to fit them.

As your organization becomes more and more cloud native, you will want to start with people and a secure foundation for individual projects to execute on.

- Invest in people. Hire smart, train them, and allow them to become experts on your deployment platforms. When you transition employees with traditional skills to build cloud-native projects, don't force them to split their time. Let them focus.
- Your scaffold will be similar to the ones we recommend for data center transformation, but you should plan on different network and security architectures. In many cloud-native deployments there is no customer-managed network.
  - Rely more on object storage (such as S3), service endpoints, API gateways, and other tools which don't require managing IP addresses for shared services. That said, you will always still need some virtual networks and a transit gateway.
- Set standards for your container networks and integrate them into your overall network management. Publish guidelines and even templates to build an easy path for independent teams to follow. Container networks can be easy to lose track of, especially when they are self-contained.
- Continuous integration and infrastructure as code are your friends. Develop supported templates for different patterns (*e.g.,* serverless, containers, standard virtual networks) which integrate your monitoring, logging, management, and security tools. Project teams can build these into their own templates; offering an easy path again helps encourage compliance.
- You will need to continuously monitor and enforce standards across hundreds or even thousands of cloud accounts. Build this early and automate provisioning through infrastructure as code and other automation capabilities.

As a final reminder, cloud-native architectures and operations are very different. Your core skills and objectives are the same, but the implementation details are incredibly different and often don't even translate between cloud providers. Providers launch new features and services on a daily basis, further challenging overworked security and operations teams.

Learn, take your time, work well with project teams, be nimble, and if you are in management... give your people time to keep up with the rapid rate of change.

If you have any questions on this topic, or want to discuss your situation specifically, feel free to send us a note at info@securosis.com.

About the Analyst

Rich Mogull, Analyst and CEO

Rich has twenty years experience in information security, physical security, and risk management. These days he specializes in cloud security and DevSecOps, having started working hands-on in cloud nearly 10 years ago. He is also the principle course designer of the Cloud Security Alliance training class, primary author of the latest version of the CSA Security Guidance, and actively works on developing hands-on cloud security techniques. Prior to founding Securosis, Rich was a Research Vice President at Gartner on the security team. Prior to his seven years at Gartner, Rich worked as an independent consultant, web application developer, software development manager at the University of Colorado, and systems and network administrator.

About Securosis

Securosis, LLC is an independent research and analysis firm dedicated to thought leadership, objectivity, and transparency. Our analysts have all held executive level positions and are dedicated to providing high-value, pragmatic advisory services. Our services include:

- **Primary research publishing**: We currently release the vast majority of our research for free through our blog, and archive it in our Research Library. Most of these research documents can be sponsored for distribution on an annual basis. All published materials and presentations meet our strict objectivity requirements and conform to our Totally Transparent Research policy.
- **Cloud Security Training and Advisory Services**: Securosis built the Cloud Security Alliance's CCSK and Advanced Cloud Security Practitioner training programs. We also provide custom trainings tuned to your needs. Securosis is a premier Cloud Security Alliance Training Partner and provides in-depth strategic and technical assessments and advisory services. We also support a limited number of members in our Cloud Security Coaching program.
- **Retainer services for vendors**: Although we will accept briefings from anyone, some vendors opt for a tighter, ongoing relationship. We offer a number of flexible retainer packages. Services available as part of a retainer package include market and product analysis and strategy, technology guidance, product evaluation, and merger and acquisition assessment. Even with paid clients, we maintain our strict objectivity and confidentiality requirements. More information on our retainer services (PDF) is available.
- **External speaking and editorial**: Securosis analysts frequently speak at industry events, give online presentations, and write and/or speak for a variety of publications and media.
- **Other expert services**: Securosis analysts are available for other services as well, including Strategic Advisory Days, Strategy Consulting engagements, and Investor Services. These tend to be customized to meet a client's particular requirements.