www.skytap.com

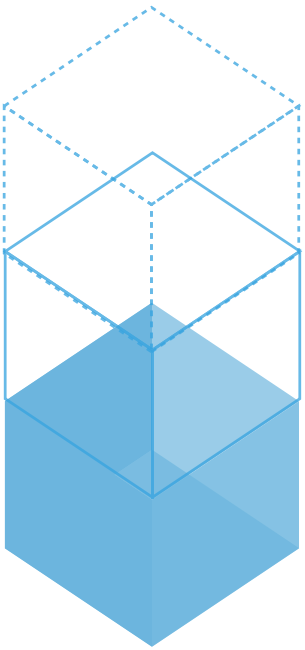
Skytap

Blueprint for migrating on-premises applications to the cloud

A cloud migration model for the enterprise

Executive Summary

The purpose of this blueprint is to describe how Skytap can help accelerate the migration of traditional on-premises applications to the cloud, whether IBM Cloud or Microsoft Azure. Skytap's capabilities align with digital modernization efforts and promote the transition to enterprise agile.



Using Skytap provides the following immediate business benefits:

- Provides a safe path to the cloud by minimizing migration re-engineering risk
- Increases developer efficiency by reducing or eliminating wait time for resources
- Leverages existing IT skillsets and avoids central cloud team failures

Skytap allows enterprises to unlock cloud benefits faster and begin innovating across applications sooner by providing the ability to take advantage of and integrate with the breadth of cloud-native services. Skytap simplifies the migration of existing systems to the cloud by eliminating the initial need to rearchitect or re-platform. The concepts outlined in this document create a quick and safer path to cloud innovation, while simultaneously reducing the reliance on existing on-prem resources.

This blueprint promotes a three-phased approach to migrating applications to the cloud. The three phases are analogous to an “application migration factory” that is used to initiate the shift from on-prem to cloud-native.

Phase 1

Leverage a lift-and-shift approach where on-prem applications take advantage of Skytap’s “run as-is” model. Skytap can reproduce traditional application architectures without requiring refactoring or rearchitecting.

Phase 2

Once application components are migrated, they can be used as a dev/test sandbox, while incrementally refactoring to native cloud services using proven design patterns such as Sidecar or Strangler. Even without using cloud-native services, incremental automation for environment builds can be introduced, starting with a non-automated working model and gradually creating a fully automated version. As an alternative, the application can simply be re-hosted (The 5Rs of Rationalization) without significantly changing its original on-prem structure.

The outlined approach maximizes the skill sets of existing talent, while delivering environment cloning, capacity on-demand, and self-service provisioning to on-prem applications.

Phase 3

Most of the application exists in cloud-native format but maintains a dedicated connection back to on-prem IT. While applications complete their transformation at Phase 3, other applications on the transformation target list are simultaneously in Phase 1 and 2.

Summary

The following document describes an efficient and incremental approach to migrating existing on-prem applications in a way that promotes a low-risk, safe path to the cloud. It supports the concept of making small and quick changes to existing applications as they are transformed from on-prem to being truly cloud-native. On-premises infrastructure does not support quick iterations and is not co-located with cloud-native resources. These together add complexity and risk to precious development cycles.

Skytap’s on-demand environments make rapid iteration easy, allowing organizations to transform applications incrementally instead of re-engineering them from scratch.

Incrementally migrating on-prem applications to native cloud services

Phase 1 – Lift and Shift

This section describes a general approach of using an application migration factory to begin the shift from on-prem to cloud-native. An initial strategy of using lift-and-shift offers maximum flexibility and allows multiple agile groups to simultaneously innovate without creating excess wait times trying to share limited resources.

1. **Initial lift and shift of existing on-prem applications in their native configuration**
   The recommended approach is to first create a representation of the existing system in the cloud, without re-engineering any components into cloud-native equivalents. The same number of VMs/LPARs, memory/disk/CPU allocations, file system structures, IP addresses, hostnames, and network subnets are created in the cloud, representing as closely as possible a clone of the system of record that exists on-prem.

   The benefit of this approach is the ability to apply cloud flexibility to an historically "cloud stubborn" system. Fast cloning, ephemeral longevity, software-defined networking, and API automation can all be applied to the initially migrated application running in the cloud.

   After this step is complete, environment clones can be handed out to multiple transformation groups, allowing each group to work independently at an accelerated pace.

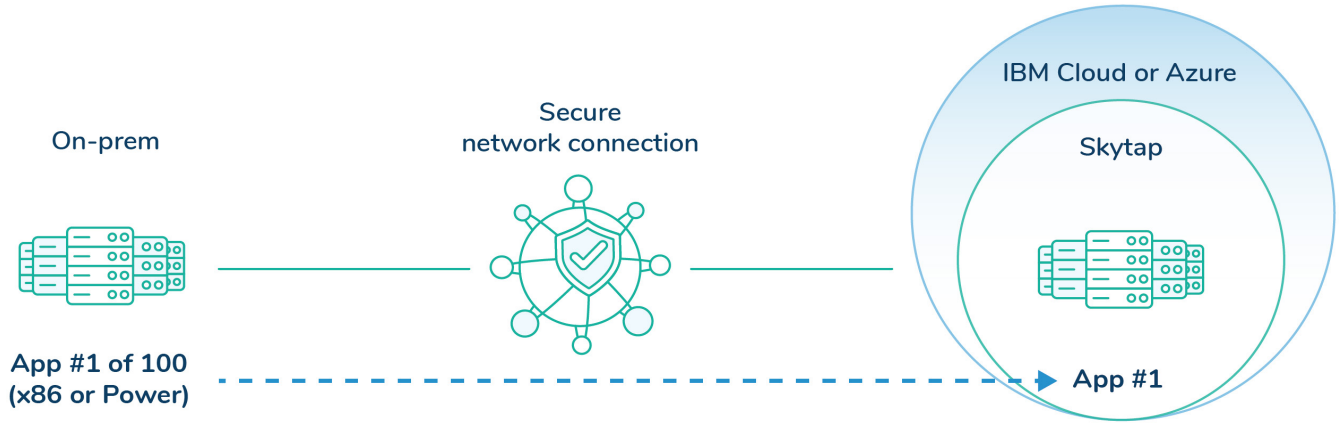


Figure 1: Phase 1 - Migrate application from on-prem to Skytap, making limited architectural changes.

2. **Save environments to templates**
Once a collection of VMs/LPARs representing an “environment” has been created in the cloud, the environment is saved as a single object called a template. The template is used to clone other working environments. Clones are exact duplicates of the template, down to the hostname, IP address, subnet, and disk allocations. In Skytap, multiple environment clones can be running simultaneously without colliding.

Creating ready-to-use environments from a template is the most powerful component of the cloud-based approach. It provides multiple exact copies of the reference system to be handed out to numerous engineering/dev/test groups, all of which can be running in parallel. There is no need to change the IP address of individual servers or their hostnames. Each environment runs in a virtual data center in harmony with the others. If environments need to communicate to other on-prem resources, they are differentiated via an isolated NAT mechanism, as described below. Many of the environments contain the same VM clone base image(s) with the same hostnames, IP addresses, and so forth.

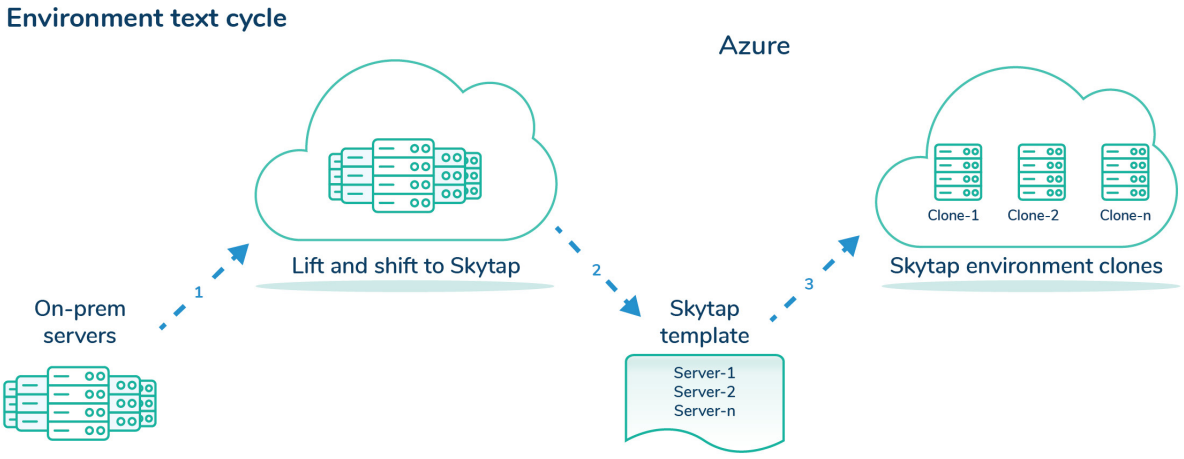


Figure 2: On-prem to cloud workflow: apps become templates that can be cloned

3. **Assign templates to Skytap projects**
Once templates are created, they are assigned in the Skytap portal to a project. Projects are then assigned to groups of users. Users can only see or access environments that have been assigned to them via the project mechanism. A QA user cannot see an environment solely assigned to ENG, for example. Skytap provides a built-in access/security model so users only see components assigned to them via the project mechanism. Users also have role assignments that allow them to view/edit/admin VMs/LPARs defined in an environment assigned to a project. The Skytap portal provides a complete and audited access control mechanism.

Creating cloned environments with duplicate address spaces

As stated previously, there are many benefits associated with creating multiple working environments that replicate the same network topology as the final target system. In this context, “replicate” means re-using the same host-names, IP addresses, and subnets within each environment.

To achieve this, some form of isolation must be implemented to avoid collision across duplicate environments. Within Skytap, each environment exists within its own software-defined networking space not visible to other environments that are also running. Using this mechanism, it is possible to create exact clones of multi-VM architectures with multiple subnets containing replicated address spaces. Each environment becomes a virtual private data center.

Cloned environments communicate back to upstream on-prem resources via a single focal point called an “environment virtual router” (EVR). The EVR hides the lower VMs containing duplicate host-names and IP addresses and exposes a unique IP address to the greater on-prem network. Using this mechanism creates a simplified and elegant way for multiple duplicate environments to exist in harmony without breaking basic network constructs. By allowing duplicate hostnames and IP addresses to exist, individual hosts do not have to go through a “re-IP” process, which is error-prone and time-consuming. The EVR paired with a “jump-host” can be configured to forward SSH requests (via SSH proxy, OpenSSH 7.x and higher), which allows SSH into each unique host in an environment. From on-prem, users would SSH to any host in the environment (e.g. ssh user@environment-1-host-2), which exposes a unique IP address to on-prem, and then relays down to the VM within an individual environment.

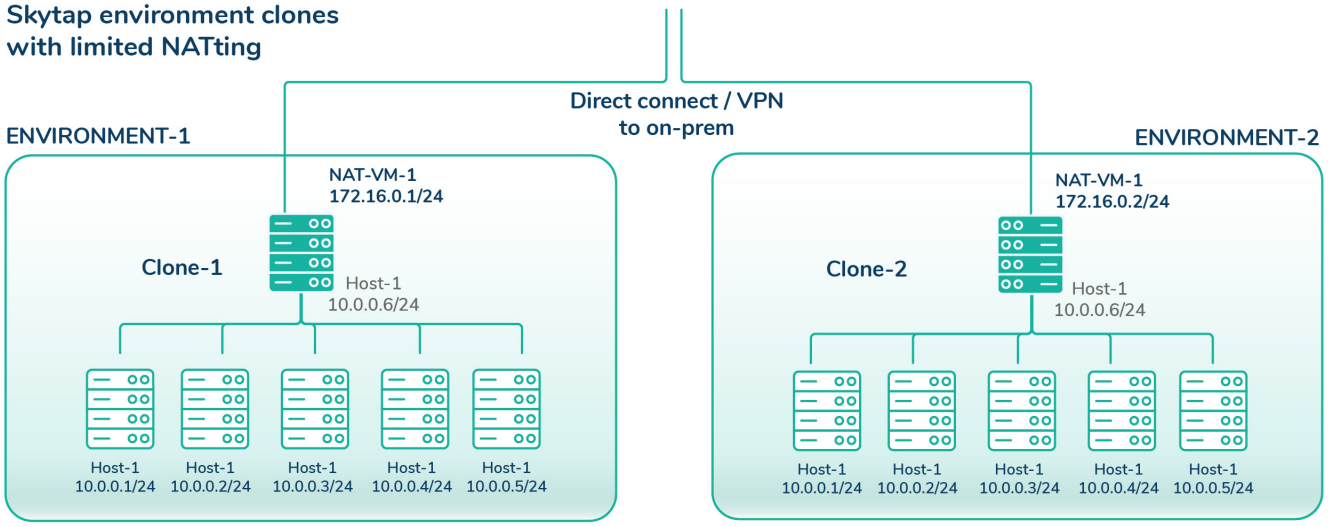


Figure 3: Multiple environments with duplicated RFC 1918 IP addresses with no collision

Phase 2 - Replatform / Refactor

As stated in the previous section, an initial lift-and-shift to the cloud without making significant architectural changes offers these benefits to agile teams who will ultimately refactor the application:

1. Allows for an incremental approach to transformation instead of starting from scratch. Refactoring R&D is done in a rationalized manner so that the overall application continues to run. This method prevents the creation of totally "net-new" development efforts that are application-wide in scope. A "start from scratch" approach across multiple applications traveling through the migration factory is high risk and seems to go against agile principles of "limit work in progress."

2. Increases agile innovation velocity by leveraging cloud capabilities such as environment cloning that can now be applied to the original on-prem version of the application. By cloning the original on-prem application, complete working versions of the reference application can be delivered to agile teams performing short duration sprints, each investigating different aspects of the application to be refactored.

   Splitting up the entire application into smaller parts lowers the risk for the entire project and will potentially shorten the overall migration. Delivering a small part of the refactoring as opposed to the entire application aligns with the agile manifesto values of "working software" and "responding to change."

3. Creates a hybrid approach to transformation that reduces organization risk when attempting to transform a large number of applications over a short period of time.

The incremental approach promoted in this blueprint follows the Strangler pattern methodology described by Martin Fowler and Microsoft. The conceptual Strangler process is visualized as:

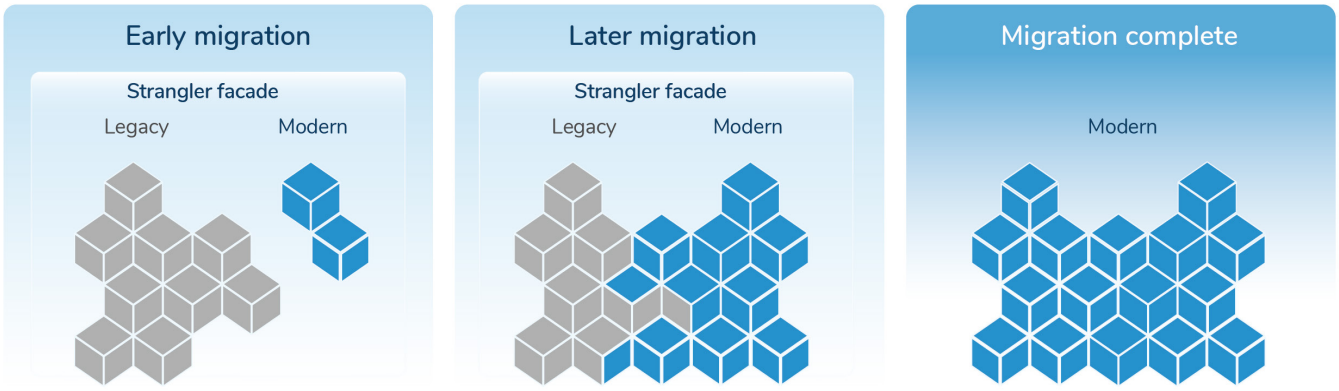


Figure 4: The balance of legacy vs. code-native shifts over migration phases

And when applied to an application moving from on-prem initially to Skytap, the diagram initially becomes:

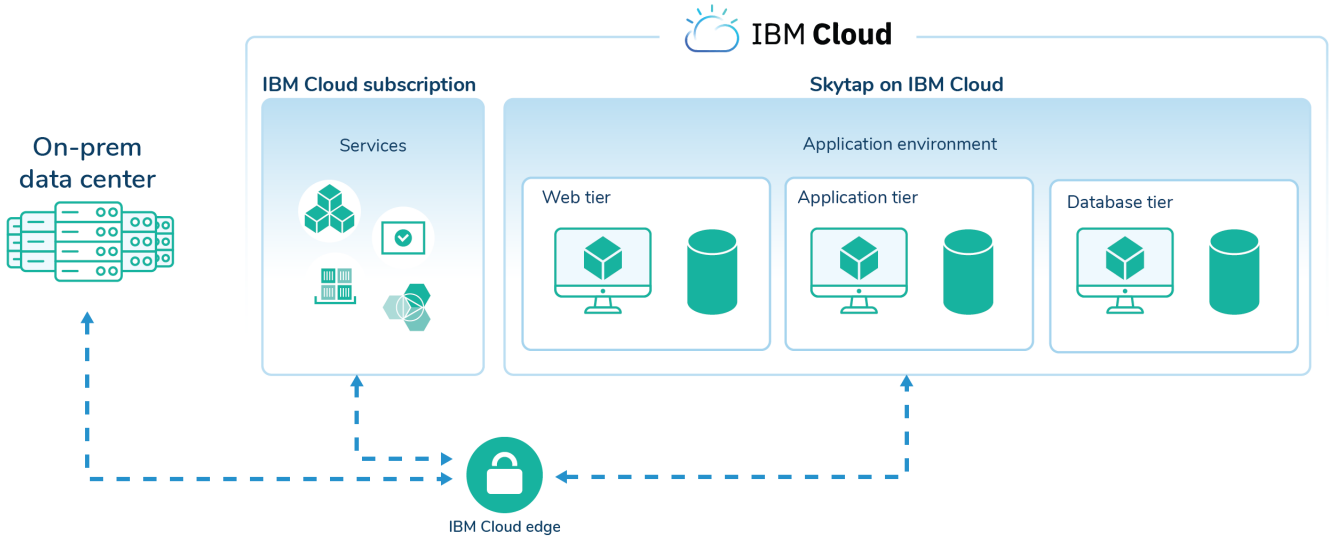


Figure 5: On-prem application initially moves "as-is" to Skytap (shown here as an IBM Cloud deployment).

Phase 3 - Migration Complete

In the application migration factory model, multiple applications are traveling through the assembly line as part of the transformation process. While applications exit the factory at Phase 3, other applications on the transformation target list are entering Phase 1 and Phase 2. As experience grows working with cloud-native services and components, the "factory floor" can speed up since solutions to problems discovered earlier in the transformation journey can be applied quickly without requiring excessive R&D, trial-and-error, and re-work.

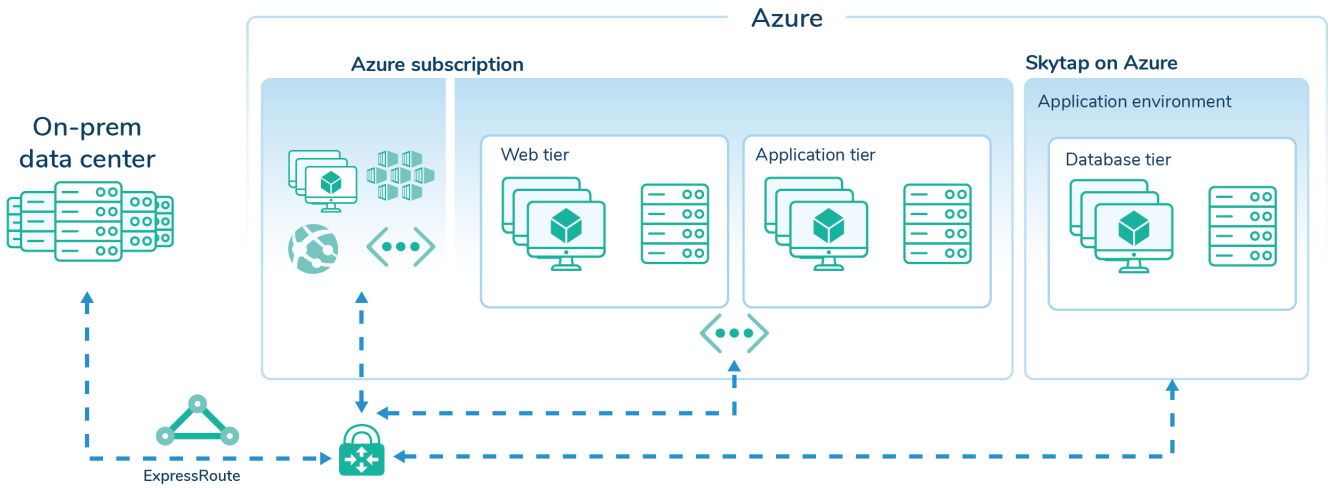


Figure 6: After refactoring, many/most application components are now native cloud services (shown here as an Azure deployment).

Reducing wait times

With the concepts and workflow of the migration factory in place, three questions can now be answered. These questions are some of the largest contributing factors to wait time, one of the "seven wastes" referenced in modern Lean.

1. *How do we create multiple isolated test environments identical down to hostnames and IP addresses?*

   QA environments "Test1", "Test2", and "Test3" should all have a host called "Database" with an IP address of 192.168.0.1. How can identical R&D and test systems exist without colliding with each other? How can these identical systems also communicate with shared resources without colliding at the network layer? Traditional enterprise thinking would require that hosts go through a "re-IP" process so that no IP addresses are duplicated on the same network address space. The downside of this approach is that the test systems are no longer exact duplicates. There is a greater possibility of having multiple configuration problems that need to be debugged by hand. The cloud model offers "environment cloning," where multiple environments with the same network topology can exist in harmony.

2. *How do we reset a test or R&D environment that has become corrupt?*

   Once an application environment has been used extensively, the test data may become stale, or automated test cases have created data that must be reset or removed before the next test run can be executed. Another variation is that multiple, but slightly different test datasets are needed to validate all configurations of the target system. Traditional enterprise thinking would use scripts and possibly other automation techniques to delete database data, reset configuration files, remove log files, etc. Each of these types of options can be time-consuming and error-prone. The cloud approach leverages "image templates" where complete, ready-to-use VMs along with their network topology and data are saved into templates. If a database becomes corrupt, or heavily modified from the previous testing, instead of resetting the data via scripts, the cloud model replaces the entire database VM and all of its data in one step. The reset process can often be done in seconds or minutes versus hours or days. Complete ready-to-use environments containing dozens of VMs can be saved as templates and reconstituted in very short amounts of time. For example, instead of taking weeks or months to rebuild a complex multi-VM/LPAR from scratch, what if it only took a few hours?

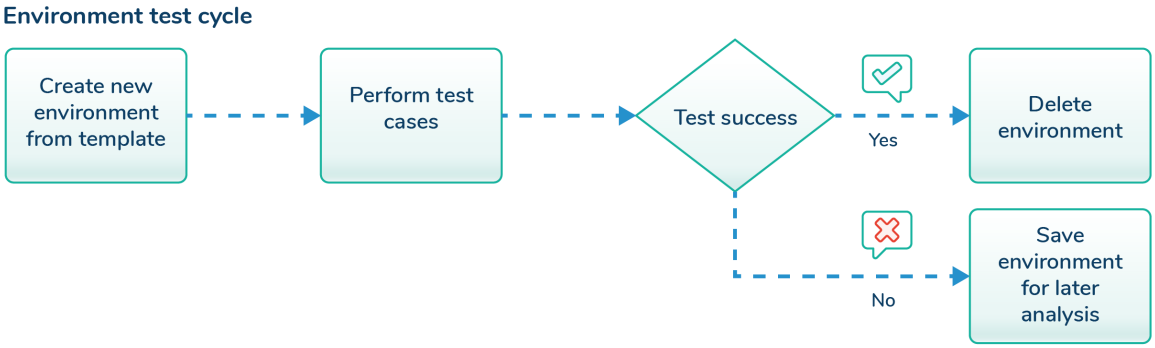


Figure 7: Environment test cycle in Skytap

3. *How do we provide self-service so that a project sprint requiring a new, clean environment is not delayed weeks or more waiting for an environment to become available?*

   Traditional enterprise thinking would limit control of cloud resources so that only a select few have direct access to the cloud resources, and those few then create environments for others. The cloud approach delivers self-service with IT control and oversight. Users and groups are given direct access to the cloud but have restrictions that limit their consumption. The overall system is protected from a runaway script that incurs excessive charges or consumes all available resources. Users and groups have a quota that limits the amount of consumption possible at any one time. Users are assigned projects that define what environments they can see. A QA user sees environments "QA1" and "QA2" while a developer might see both of those as well as "R&D1" and "R&D2". Finally, the cloud system provides universal auditing so that user activities are tracked and available for reporting. The question of "Who deleted that AIX LPAR?" is no longer a mystery.

   Self-service increases velocity, thereby increasing the potential for greater innovation. The concepts of increasing velocity and reducing wait times and bureaucracy are described in Doing Agile Right when describing the qualities of an agile enterprise.

Technical use case taxonomy for agile teams participating in transformation

The following use cases apply to many server transformation projects where different groups of participants need access to representations of the final target system or some subset of the final design. The key is to quickly deliver the infrastructure needed to perform a specific need or task. Waiting weeks for infrastructure delivery should be considered an "anti-pattern" since the cumulative time of waiting over the project's course would be considerable. Building internal resource delivery processes with slow delivery times goes against the concepts described in works such as The Phoenix Project, The Goal, and Healthcare Digital Transformation.

R&D sandbox

Developers need a way to create their own "environment" of components that represents the target system. See Gene Kim's reference to "access to production-like environments" for developers in *The Phoenix Project*. Providing "representative" environments instead of mock environments running on local workstations or laptops would be a key indicator of the ability to reliably prove out many architectural assumptions being made in the project.

Classical QA automation testing

Instead of QA having to share a limited number of environments that commonly develop "configuration drift" for the current versions of components, QA should be able to easily create 'n' number of QA environments "QA-1", "QA-2", "QA-3", "QA-n." Leveraging cloud representations of the target system will allow QA to destroy and rebuild the correct target environment from scratch within minutes or hours. No more scripting to back-out, reset test data. No more "reset" scripts to return configurations to a starting state. The QA environment is completely ephemeral (short-lived) and may only exist for the duration of the test run. If tests fail, the entire environment is saved as a complete snapshot, aka template, and attached to the defect report to be reconstituted by ENG when diagnosing the problem. For the next test run, a completely new environment is generated from the template and is separate from the previous environment used in past test runs that may contain defects.

By leveraging cloud resources for testing, more tests can be run at potentially a faster pace and with higher quality results, hopefully reducing defects that slip through to the final product. Reducing defects is also described in "seven wastes" referenced previously.

Integration testing

Traditional enterprise thinking historically creates a limited number of integration test environments shared among many groups. Because of this, the integration environment is often broken, misconfigured, out of date, stale, or unusable in some way. Environment "drift" becomes a barrier to doing regular testing.

Applying cloud thinking to the building of on-prem systems allows for different integration testing approaches to be used in an economical and efficient manner. The cloud can create multiple integration environments that are all "identical" based on the current target goal. R&D and ENG subgroups can each have their dedicated integration environment that can combine work from multiple squads of the same discipline, without colliding with other system components. For example, all the teams working on database changes can first integrate their work into a localized integration testing environment. Once successful, they can move the bulk modifications to the higher level where all system components are being combined.

Chaos engineering, "What if we take server X offline?"

Chaos Engineering for Traditional Applications documents the justification and use cases where cloud-native chaos engineering theory can be applied to traditional applications running on-prem. The idea is that a clone matching the original system of record can be re-created in the cloud. Destructive testing can be performed on the cloud-based placeholder and then the learned results can be applied to the on-prem system. A new level of "What if XYZ happens?" can be achieved by leveraging a cloud-based proxy for the original on-prem system. Some categories of problem areas that need random failure testing would include:

Resource-based, such as:

- Low memory
- Not enough CPU
- Full disk volumes
- Low network bandwidth, high latency

Hardware failures like:

- Failed disk drive
- Failed server
- Disconnected network

Not-so-obvious ones could be:

- Database/server process down
- Microservice down
- Application code failure
- Expired certificate(s)

And even less obvious: is there sufficient monitoring, and have alarms been validated?

Each category of items requires the execution of multiple "experiments" to understand how the overall system reacts when a chaotic event is introduced into the system. The cloud can then be used to recover the system back to a stable state. Quick environment recovery allows for the execution of multiple, potentially destructive experiments.

Disaster recovery

The traditional DR model creates recovery infrastructure in an alternate location available for use in the event that the on-prem primary system becomes unavailable. With the ability to host IBM Power workloads in the cloud, a new DR model becomes viable that extends beyond x86 workloads.

It is now possible to host both IBM AIX and IBM i (iSeries) in the cloud. The cloud now becomes the DR location for workloads that historically had no path to the cloud. The ultimate viability of this approach will vary based on complexity, scale, and performance but it is now a legitimate option that can be considered. The most interesting aspect of this approach is that the single DR workload can be "mixed" across x86, AIX, IBM i, all running in the same cloud-based DR data center. This creates the ability to model the original on-prem system of record in a way that does not force re-architecture or redesign. The original system of record can be recreated in the cloud with minimal changes to its original design.

Packaged applications

Many organizations rely on packaged applications in order to run their core business activities, such as Epic for healthcare, Oracle Financials, Lawson, and others. These applications go through a configuration and/or tailoring process for each organization. The process of tailoring these applications requires test instances and complete environments matching the software configuration of the final production system.

A tremendous opportunity exists to run pre-production systems in the cloud rather than on-prem. In many cases, it is not necessary to exactly match performance characteristics of the on-prem system in order to validate the configuration process of these packaged applications. A scaled-down version of production in terms of hardware resources, but matching exactly the configuration parameters of the production application is a model that could be potentially leveraged in the cloud. In many cases, the pre-production environments in the cloud can be temporary for the duration of a specific application release. This allows for economic efficiency versus standing up permanent resources on-prem.

Summary

This document outlines an approach that can be used to provide a low-risk, safe path for the migration and transformation from on-prem to the cloud. The intermediate use of Skytap as part of the transformation process allows for an incremental approach to the migration. The ability to experiment with the design, perform greater amounts of QA testing, and execute advanced testing concepts like those described as "chaos engineering" all create a safe path to the cloud.

The outlined approach unblocks traditional enterprise migration models by giving production-like environments to all groups that need them as outlined by Gene Kim, and eliminates the constraint of environment resources as described in The Goal. The transformation model described here is the solution to the anti-patterns historically created on-prem where agile sprint teams and squads wait weeks or months for needed environments. Skytap provides a self-service, on-demand approach to delivering environments in minutes or hours instead of days, weeks, or months. Skytap supports Power alongside x86 in both IBM Cloud and Azure and complements a digital transformation strategy.

Skytap

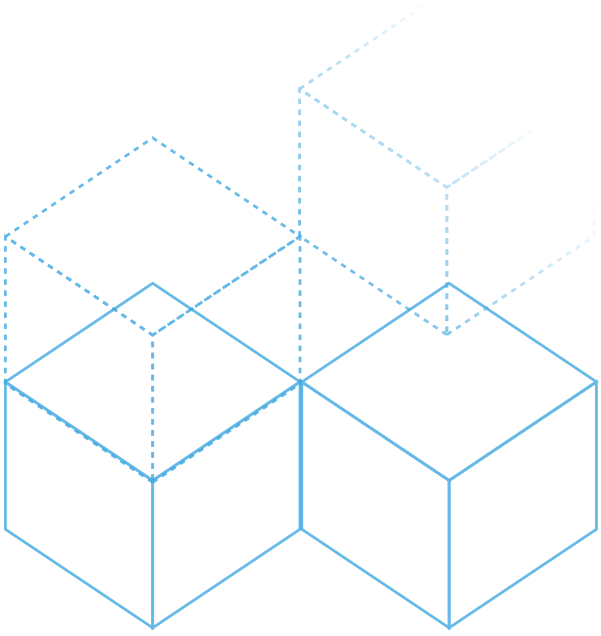
About Skytap

Skytap is a cloud service purpose-built to natively run traditional systems in the cloud. Our customers use Skytap for running production, disaster recovery, virtual training labs, and development workloads. We are the only cloud service to support AIX, IBM i, and Linux on IBM Power together with x86 workloads, enabling businesses to accelerate their journey to the cloud and increase innovation. To learn more about Skytap or schedule a demo, visit www.skytap.com

Skytap Headquarters
255 S King St, Suite 800
Seattle, WA 98104
206-866-1162

Skytap Canada
1 University Ave, Suite 3139
Toronto, ON M5J 2P1
888-759-8278

Skytap EMEA
30 Stamford Street
London SE1 9LQ
888-759-8278



twitter.com/Skytap

linkedin.com/company/skytap/