No App Left Behind: A Practical Guide to Using Skytap Cloud, Containers & CI/CD Tools for Modernizing Traditional Applications
# Table of Contents

**Executive Summary** .................................................................................................................. 3

**Skytap Cloud and Traditional Application Modernization** ....................................................... 4

**How Can Skytap + Containers Accelerate Traditional Application Modernization?** ................. 5

**Using Skytap Cloud and Containers to Modernize Infrastructure** ............................................. 7

**Skytap + Containers Working Together During Architecture Modernization** .......................... 12

**Summary: Skytap Environments Add Value at All Stages of an Application Modernization Journey** ............................................................................................................. 16

**Additional Resources** ................................................................................................................. 17
Following our introductory white paper which introduces the concepts and considerations associated with containers, this guide will address how you can use Skytap Cloud and Containers together to accelerate modernization for your traditional applications.

There is no dispute that containers are a rising technology that is seeing wider adoption in the enterprise. At Skytap we recognize both the value and the complexity associated with this technology and have built capabilities into our cloud that accelerate the move to containers as part of the modernization journey. For traditional applications, that journey varies across enterprises based on their knowledge of cloud capabilities, their adaptability to change, the adoption of automation, and the requirements of the application.

Skytap participates in each phase of the application’s modernization journey through our Infrastructure, Process, and Architecture Modernization (IPA) approach to successfully streamline and accelerate modernization alongside containers and other technologies as noted throughout this document. Our goal is to eliminate complexity where possible to enable teams to be more effective in developing and managing their applications with as much automation as possible—natively or by supporting integration with other tooling.

As in choosing a cloud provider, there was once a belief at that a single provider could deliver on all of a business’s needs, but that has proven to be invalid with the rise of multi-cloud adoption. A similar pattern is proving true across the underlying application technologies as well. Virtual machines, various flavors of containers and their orchestration tooling, DevOps automation tools, native cloud services, and traditional applications will all need to work together, as no singular solution on its own can address the complete application lifecycle.

Skytap accelerates traditional application modernization by working in concert with these technologies and brings unique capabilities to help enterprises accelerate time to value, reduce errors in the SDLC, and deliver true-self service that provides teams with the access they need to on-demand resources so they can innovate faster.
Skytap is a public cloud provider that accelerates enterprise innovation by modernizing a traditional application's infrastructure, processes and architecture.

When considering the requirements of a traditional application you will find that hyperscale clouds often cannot address the requirements of these applications. The table below outlines some key considerations for modernizing traditional applications in the cloud.

Further, modernizing traditional applications is challenging and requires enterprises to invest time & people. Skytap Cloud focuses on helping enterprises accelerate this journey. By migrating from on premises infrastructure to the Skytap Cloud gives enterprises the benefits of cloud scale, shift from CapEx to OpEx spend, and increased productivity for IT and Application Development teams through templates and self-service environments, plus predictable pricing through user quotas.

Skytap Cloud provides environments to run and modernize your traditional, mission-critical, x86 and AIX/Linux on Power applications. Skytap environments are self-contained virtual data centers that have everything needed to run a complex application: from network segments to fully deployed applications.

Users can easily create, clone, share, and collaborate with these environments while maintaining tight IT visibility and user quotas that give the business predictable pricing.

Introducing CI/CD tools like Jenkins, Urban Code, and others to a traditional application in a Skytap Cloud environment makes it easy for application development and QA teams to adopt agile, cloud native practices. This enables faster deployment through automation & centralized build servers, faster release cadence & improved code quality by running tests on parallel copies of an environment.

Skytap Cloud combines proprietary self-service, fast provisioning, user quotas and environments which contain traditional applications, modern CI/CD tooling and containers to accelerate the time realize the value in cloud native, microservices based applications.

**Compare Clouds for Modernizing Traditional Applications**
**How Can Skytap + Containers Accelerate Traditional Application Modernization?**

There is no doubt that containerization is a new paradigm that is becoming more widely used throughout the enterprise. In most cases, enterprises are predominately virtualized and starting to consider containers as part of their modernization journey. Skytap Cloud can run Docker containers & Kubernetes-backed platforms alongside traditional applications in a single Skytap environment.

At Skytap, we approach application modernization in three phases: Infrastructure, Process, and Architecture (IPA).

During Infrastructure Modernization assets are typically moved into Skytap “as is,” which is often a VM. Customers may be using containers on-premises and can deploy them into Skytap. In this phase, teams are getting comfortable with new cloud paradigms while keeping a familiar operating and development approach to the application.

<table>
<thead>
<tr>
<th>Skytap</th>
<th>Other Clouds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose built for running Traditional Applications in the cloud</td>
<td>APPLICATION FOCUS: Built for developing new cloud native apps and services</td>
</tr>
<tr>
<td>Simple, Powerful UI and Role assignment for self-service</td>
<td>USABILITY: Complex UI, Roles and Services requires deep knowledge</td>
</tr>
<tr>
<td>Migrate effectively with L2 Networking Configuration Capture</td>
<td>MIGRATION: Refactor &amp; Reconfiguration often required to successfully migrate</td>
</tr>
<tr>
<td>Simple Configuration - 1 Environment per App, All Configs</td>
<td>COMPLEXITY: Many Vms, Services, &amp; Interdependencies = Complexity</td>
</tr>
<tr>
<td>Manage Costs with Scheduling and AutoSuspend of Resources</td>
<td>COSTS MANAGEMENT: unexpected cost overruns - reactive/predictive cost mgmt.</td>
</tr>
<tr>
<td>Unlimited Bandwidth</td>
<td>BANDWIDTH COSTS: $$ per gig</td>
</tr>
<tr>
<td>AIX on Power</td>
<td>BAREMETAL AIX ON: not available</td>
</tr>
<tr>
<td>Linux on Power</td>
<td>BAREMETAL LINUX ON POWER: not available</td>
</tr>
<tr>
<td>Integrate with Best of Breed Cloud Native Services</td>
<td>MULITCLOUD CHOICE: Lock-in and deploy on 1 cloud for all services, not always best of breed</td>
</tr>
<tr>
<td>Deployment Choice - Single Tenant, Dedicated Region and Multitenant</td>
<td>DEPLOYMENT OPTIONS: Predominantly Multitenant</td>
</tr>
</tbody>
</table>
During Process Modernization, organizations are focusing on improving their SDLC from both a people and technology automation perspective. Here, they are shifting to Agile, CI/CD and Configuration Management tooling such as Jenkins, UrbanCode, Chef, Puppet, Ansible and others with a focus on minimizing errors across the SDLC so that they can innovate faster. This will prepare them for developing in a cloud native world that includes microservices, containers, and cloud native service integrations.

During Architecture Modernization, organizations are now comfortable with the cloud and are prepared to make changes to the underlying application components. This means starting to decouple the monolith and break it into smaller, more agile components that can be developed and released independently, potentially reducing the footprint of the application. In some cases, the entire app with be rearchitected, in most cases, the application will continue to be hybrid with components that will continue to run virtualized while others will shift to containers and microservices. All of which can run side by side or be integrated in Skytap Cloud.

Let's discuss how Skytap and containers can work together to accelerate application modernization.

**When using containers there are required components that must be in place:**

1. A container must have an image.
2. A container should have an associated registry to manage images and distribution at scale.
3. A container must have a host.

Skytap can assist with all three of these items and more to accelerate the adoption of containers in an enterprise while also furthering the applications’ modernization journey. This isn’t an either/or—rather it’s two technologies working together to enable an organization to more seamlessly move through modernization in the simplest manner. In this last section we'll outline how Skytap works with Containers in the three stages of IPA.
In this section we’ll demonstrate how Skytap Cloud can seamlessly integrate the components required for containers to work in the cloud as noted above.

**Use Skytap Cloud as the Host for Containers**

In this first example, Skytap is shown as the Host for the Container. This example assumes that the organization may be building the container images on developer workstations and that a Repository is already in place. Skytap therefore becomes the cloud where the Container Host resides and the Container runs. The Host has complete access to all functions in Skytap and can be interacted with via command line or the Skytap UI.

The container will have the ability to communicate with any other VMs that make up the traditional application. Often, customers will require a combination of VMs + containers as not all components will be able to shift to containers without some rewriting or refactoring.

For this reason, customers typically progress to containers during architecture modernization rather than the first step in their journey.

**Using Skytap and Docker Together - Registry Hosted Outside of Skytap**

1. Developers will need to create the Docker config file and the image to get started. In this example a local computer is used, which ultimately will become limited in resources. Developers often execute commands such as ‘run’ to test the container and its image before making it available in the registry. Once complete, they ‘push’ it to the registry.

2. Containers require a Registry. The Registry keeps track of all of the image files and is the location where images are sourced from. It can be stored on-premise but is more commonly hosted in the cloud.

3. A container is required to have a Host. That host is typically a virtual machine. When using containers in Skytap, you can easily create your VM host and attach any number of containers to it. You will gain all the benefits of Skytap natively when doing this - cloning, templates, copy to region, quotas, and more.
Use Skytap Cloud to Develop Images, Home the Registry, Host the Containers and Their Runtime

In the second example, Skytap can be effectively used across all three areas by providing elastic resources that can grow with the organization.

1. Skytap can provide developers with predefined, cloneable environments for development and test thus reducing resource limitations on their local machines.
2. Skytap can be used to home the container registry and make it accessible to everyone.
3. Skytap can be the Host for containers that are used to modernize components of the traditional application when ready.

![Diagram showing development, registry, and container flow in Skytap]

Using Skytap and Docker Together
Developer VM, Registry and Container Host inside of Skytap

1. Developers will need to create the Docker config file and the image to get started. In this example, a VM image is made available for developers in Skytap. This image can be easily templated and copied for others to use in round the sun development. This approach reduces errors in development. Developers can still execute commands such as ‘run’ to test the container and its image before making it available in the registry. Once complete, they ‘push’ it to the registry.

2. Containers require a Registry. The Registry keeps track of all of the image files and is the location where images are sourced from. Here, the registry is hosted in Skytap rather than on-premise or externally to simplify management. It too, could be templatized. ICNR is used for intercommunications.

3. Similar to Example 1, A container is required to have a Host. That host is typically a virtual machine. When using containers in Skytap, you can easily create your VM host and attach any number of containers to it. You will gain all the benefits of Skytap natively when doing this - cloning, templates, copy to region, quotas, and more.
In this example, if developers want use their physical computers for development that is possible. The flow above would not change however you would not create ENV1.

The previous examples demonstrate how Skytap and containers can seamlessly work together at the infrastructure layer. To best take advantage of the cloud, enterprises need to adopt agile methodologies and tools. Skytap Cloud makes this easy with integration partners like Chef, Puppet, Ansible, UrbanCode, Vagrant, and many other CI/CD, automation, and DevOps tools.

Enterprises can use existing tools when they migrate their traditional application to Skytap. Fast cloning of Skytap environments allows teams evaluate new tools in isolation from ongoing development and we provide a number of public templates that can be added to your traditional application running in Skytap. With Skytap environments comprised of VMs and containers, enterprises can adopt a single set of tools to run across both, further aiding the modernization process. Further, adopted tools running in Skytap can be saved as templates and shared across teams to provide consistency.

“Organizations that have successfully adopted DevOps are better equipped to handle the technological challenges of containerization because they have already tackled the cultural transformation that is also necessary.”

- Gartner Docker report, November 2017

**Using Skytap Cloud During Process Modernization to Prepare for Containers In The SDLC**

Let’s discuss how using Skytap with containers benefits the organization beyond the infrastructure elements and brings benefit into Process Modernization which focuses on agile, devops, CI/CD tooling and SDLA automation. Skytap can assist by providing the following benefits:

**Environments Provide Safe Experimentation and Evaluation of Orchestration Tools to Solidify Your Strategy**

Container orchestration tools such as Kubernetes are complex, and it can take significant time to retool components of an application to communicate using Kubernetes’ cluster networking.

Skytap environments facilitate safe experimentation and development of application code changes needed to accommodate Kubernetes. Developers can start with an instance of their traditional application in a Skytap environment, then install Kubernetes inside it and experiment with application code changes to use cluster networking—safely. If the application
breaks, a developer can simply delete the environment and create a new one from the known starting point. No one is adversely affected.

Contrast this to attempts to install and interface with Kubernetes on shared non-Skytap staging environments. Either an entirely new staging environment will need to be built for the containerization effort (and it may still break frequently), or else developers make changes on a shared pre-production environment, potentially interfering with or even blocking the work of their peers. This may result in slower delivery of application updates to customers, because containerization work interferes at an infrastructure level with the ongoing development against the traditional application.

**Compatibility with all container orchestration and clustering technologies**

Skytap environments are compatible with all container orchestration and clustering tools (e.g., Kubernetes, Docker Swarm) and in fact make it easy to evaluate them side by side. Developers can simply create multiple Skytap environments—containing the traditional application they want to containerize—and install the container orchestration technology of choice in each. Environments are isolated from one another, and each acts as a sandbox in which to observe how the application behaves in situ with each orchestration technology. Ultimately, this accelerates the decision of which orchestration technology is best-suited for the application. It gives developers evidence and confidence that their decision is sound.

**Fast self-service environments and templates built from configuration management tools to accelerate the SDLC**

Quick cloning of Skytap environments complements configuration management tools like Chef and Puppet. Configuration management tools build an application by executing a sequence of rules or steps—for instance, downloading packages from the internet, compiling software, creating artifacts from configuration files. These tools solve a real problem, but in practice can occasionally run into errors and often take hours to build the application from scratch.

Using Skytap, a user or even an automatic process can invoke a configuration management tool like Chef to build up a working instance of the application. Once the application has built successfully, it can be saved as a Skytap template. Then, other users of Skytap can create an environment from this template in seconds. Their environment is guaranteed to be a bit-for-bit identical copy of the original environment, and therefore their application is guaranteed to be in a known good state. Perhaps as important, there are significant time savings; while using the configuration management tool to build the environment may take hours, once the known good state has been captured in a Skytap template users can acquire their own copy in mere seconds.

**Configuration data surrounding the application**

At runtime, even a fully containerized application is situated in an environment where additional data surrounds it. For instance, there are environment variables, service discovery, secrets management, etc. that aren’t encapsulated in the container image. This surrounding data is “married” with the container at deployment time, and will likely vary between environments (dev,
test, staging, pre-production, production). Skytap can help by providing pre-configured environments with the correct configuration data. This eliminates the guesswork for correctly configuring the application, improving speed and agility.

**Skytap environment cloning extends quality assurance coverage**

Fast self-service provisioning of Skytap environments enables quality assurance teams to quickly clone and run multiple test environments in parallel, because Skytap clones environment's down to layer 2 network configuration each environment is an exact replica. Skytap environments run in isolation, so QA can run tests against multiple copies of the same environment in parallel, decreasing test times and increasing test coverage. As enterprises continue to modernize their application, comparative test cycles run in parallel ensure the desired outputs between the original and new state remain the same. When modernizing a traditional application in Skytap Cloud, cloning of self-service environments enables enterprises to continue to run test cycles across the traditional application, while in parallel, test individual components of the application. With Skytap, test cycles can seamlessly run in parallel without conflicts.

**Skytap Cloud's practical application of Containers, CI/CD, and Orchestration**

The approaches outlined above are exactly how Skytap's SDLC works internally (we use Puppet, Ansible, Docker, Kubernetes, etc.). A provisioning run for even a portion of Skytap's application takes hours to complete, and few members of our technical teams are truly familiar with or interested in fixing inevitable provisioning problems. So, we create and validate Skytap environments from scratch several times a day using automated tools, and we make the results available to everyone. These canonical environments might remain valid for only a day or for many weeks depending on the need. All of our technical teams can clone the specific version of a ready-to-go environment they need, without having to wait around and with confidence that the application will be in a known good state.

Further, Skytap engineering uses Skytap environments to allow developers to easily create pre-canned application environments for development. These environments come with Kubernetes pre-configured correctly for the dev use case and containers already deployed. This significantly speeds up our development, since developers can skip the manual, tedious, and error-prone process of configuring and deploying the application themselves.
Modernizing a traditional applications architecture occurs over the course of a number of phases as outlined in our IPA Framework. This approach will enable you to do some things now that will make future steps much easier when you are ready. In this section we will outline the considerations required achieve architecture modernization.

### Architecture Modernization Phases

#### Refactor
- Once Lift & Shift is complete introduce new languages to prepare for microservices.

#### Replatform
- Define Requirements for the App Behavior/Performance
- Identify Migration Pattern
- Determine Modern Host – VMs or Containers, both
- Decouple and begin to modernize the Tiers with Microservices
- Introduce Bindings & Brokers

In the table below you will see that there are phase to modernizing an application's architecture, where some of those phases start in areas previously discussed. Architecture modernization is not usually accomplished on its own unless the organization has made a conscious decision to refactor the monolith immediately. The more common route is that components are rehosted, replatformed, refactored, and rewritten over a period of time.
<table>
<thead>
<tr>
<th>Rearrchitecture Phases</th>
<th>Considerations for Modernization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Phase 1 - Move into the Cloud (Infrastructure Modernization)</strong></td>
<td></td>
</tr>
<tr>
<td>Lift &amp; Shift</td>
<td>Determine the appropriate infrastructure for the application based on its requirements – Multitenant, Dedicated Region, or Bare Metal. Lift &amp; Shift application as-is into Skytap ensuring that security, identity, and network communications are in place should the application require connectivity back to on-premises or to an external cloud to leverage cloud native services in the future. The application is typically already virtualized when this step is taken. In some cases it may be physically installed on the hardware and that is where a bare metal scenario may be applicable if the application cannot be virtualized before lifting.</td>
</tr>
<tr>
<td><strong>Phase 2 - Refactor (Process/Architecture Modernization)</strong></td>
<td></td>
</tr>
<tr>
<td>Internal Refactor</td>
<td>Componentize the internals of the application, introducing indirection and abstraction to define key functional paths and data flow in the application. Introduce and/or increase code coverage of automated tests to ensure expected application functionality is preserved. Externalize configuration from configuration files and databases; hardcoded port numbers should be configurable if possible. Introduce new approaches to the SDLC Process including Agile, Devops, and CI/CD Tooling for increased automation.</td>
</tr>
<tr>
<td><strong>Phase 3 - Replatform (Architecture Modernization)</strong></td>
<td></td>
</tr>
<tr>
<td>Decouple the Applications’</td>
<td></td>
</tr>
</tbody>
</table>
| Presentation | Depending on the application, your presentation layer may be an installed client side application, a web browser or an emulator. Below are examples of possible cloud native approaches to modernize the Presentation Tier.  
  - Modernization for client side presentation tier approaches include:  
    - Mobile Services - leverage cloud native mobile services instead of client desktop only  
    - Web Services - Shift from client based applications to cloud native web services for presentation  
  - Modernization for web based presentation tiers include:  
    - Web Services - Shift from browser dependent applications to cloud native, browser agnostic web services  
    - Mobile Services - leverage cloud native mobile services instead of client desktop only  
  - In this tier of the application, containers may serve as a mechanism for incremental modernization alongside microservices. In which case both can still be hosted and run within Skytap Cloud. |

In any of these scenarios, security and load balancing should be considered in the design. This may include application firewalls, front end load balancing, increased network capacity or resource proximity to ensure that you achieve both the security and performance required.
**Logic**

When considering modernizing the logic tier you may have to introduce concepts of messaging (service bus/queues) and caching, such as RabbitMQ mentioned previously. These principles are not new, but they way they are applied in the cloud provides better efficiency and may reduce coding required. They can improve application responsiveness related to latency. Depending on the application, your logic layer may include custom code and/or middleware. The Data row includes some examples of cloud native approaches to modernize the Data Tier. Skytap Provides a RESTful API that can be used for communication from any cloud into Skytap Cloud.

**Data**

Modernizing the data tier requires consideration for availability and security. Availability is concerned with how the application responds in erroneous situations (non-disaster). If it needs to be highly available you'll want to make sure that your design accounts for rapid replication of data and failover in the event of an issue. Security is concerned with access to the data layer by both people and the application itself. Who and what services have access to the applications? Read, write, or both?

**Determine the Deployment Format**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a. VMs + Containers</td>
<td>Virtual machines are often the first approach chosen because they are familiar to both dev and ops due to on-premises experience however they often treat everything as a single entity. Using Containers and VMs together is often the first step into modernizing through microservices. Allowing some components to be decoupled while others continue in their original state.</td>
</tr>
<tr>
<td>b. VM -&gt; Containers</td>
<td>This approach decouples the application even further to accelerate modernization and development. Adding in containers enables devops to release more quickly as the monolith is decoupled further, more components can be built and updated independently and dependencies no longer carry as great of a risk. As per above discussions, moving from a VM to a container is not a simple copy and paste, you must factor in all the other considerations as you are now starting to rearchitect how the application functions.</td>
</tr>
<tr>
<td>c. Containers Only</td>
<td>This approach decouples the application the furthest for accelerating modernization and development. However, it may not be realistic for a traditional application without completely rewriting the application components or leveraging cloud-native services that will replace the monolithic components keeping it within the legacy architecture.</td>
</tr>
</tbody>
</table>
## Choosing a Deployment

### Orchestration Platforms for Deployments at Scale

When adopting an orchestration platform for deployment, it is important to do this in conjunction with your deployment format. If choosing a container format/platform, there are many options available, including:

- Cloud Foundry
- Openshift
- IBM Cloud Private
- Kubernetes

Mixed deployment formats can live in colocation to one another on Skytap, and continued use of VMs directly, require no platform other than Skytap.

## Phase 4 - Brokering and Binding

### Service Brokers and Catalogs

After the application has been replatformed, the services that an application depends on (i.e. Datastores, third party services, etc) need to be abstracted at platform level. This is done by means of a service broker and/or service catalog. Brokers provide a means for procurement of services between the service catalog and the IaaS. The service catalog itself provides the procurement facilities for the deployment platform and the brokers.

Service catalogs are shipped with many deployment platforms, particularly container platforms. Brokers are normally available from the infrastructure provider. To ensure broadest compatibility make sure both the catalog and broker support the Open Service Broker API.

### Service Binding

After a service has been procured, it is associated with the application where configuration information related to that service is exchanged with the application. This process is called binding. It is automated between the service catalog and the deployment platform. This may or may not require changes to how you externalized the configuration of your application.
Skytap environments add value at all phases of the containerization journey, from the very beginning where nothing is yet containerized, through intermediate phases where only some components of the application are containerized, up through full containerization of the application.

If an application is hosted within a Skytap environment, at the beginning and in the intermediate containerization phases, isolation and safety are provided. Since Skytap environments are fully self-contained, the containerization effort doesn’t halt or interfere with development against the traditional application hosted somewhere else (either in a separate Skytap environment or on-premises). A company doesn’t fall into a dichotomy of adding business value via their traditional, monolithic application vs. decomposing, modernizing, and containerizing it—they can occur in parallel.

It may well be the case that the final application is hybrid, where most of it has been decomposed into containerized microservices but where there are one or more legacy pieces that should or must remain in VMs. Skytap environments are specially designed to support complex hybrid applications. Companies can rest assured that when using Skytap, their efforts to containerize their applications won’t be wasted or failures if there are pieces that resist containerization.
Additional Resources

Continuous Delivery of Fully Functional Environments: How Skytap Uses Skytap Cloud to Achieve DevOps

How Skytap Complements, Not Competes with, Puppet and Chef

Knocking Down Our Own Skytap Stack with "Jenga"

How to Prepare Your Enterprise for a Docker Containers Initiative

The State of Containers

Container Strategy for Executives