



The journey to the cloud — a technical review of Simloud

Simloud streamlines access to public cloud infrastructure

Whether supporting Application Development, QA testing, R&D Labs, DevOps performs a delicate balancing act. On the one hand, Operations needs to accelerate infrastructure provisioning for all users. On the other, it should simultaneously adhere to compliance needs, business controls, and security mandates.

To navigate this increasingly complex infrastructure stack successfully, DevOps needs to provide and manage resources for core infrastructure (compute, virtualization), containers, microservices, multi-cloud, and edge infrastructures, all of which require expertise in multiple technical domains. As a result, provisioning requests accumulate and become backlogged, with DevOps eroding the productivity of the teams they are tasked to support. But there's another way of dealing with it more efficiently.



SIMLOUD'S PLATFORM FOR DEVOPS

Simloud's Environment-as-a-Service (EaaS) platform allows DevOps to define and provision complete infrastructure stacks to support any application or Cloud Service.

Feature-packed solution

- Role-based self-service access to infrastructure removes provisioning bottlenecks, while built-in guardrails protect the use of resources.
- Environments powered by modular blueprints Simloud environments encapsulate more than core infrastructure (public cloud, virtual machines, storage, container orchestration, etc.) and include data sources, network topology, cloud services, and applications services such as Kubernetes.
- Infrastructure elements are defined in modular templates or blueprints that include all necessary elements to support an application throughout its lifecycle.
- Users create these blueprints with Simloud's YAML Infrastructure as Code (IaC) format, and/or the enhanced UI delivered as a wizard, or by importing existing IaC scripts from tools already in use by their development or infrastructure operations staff.
- The blueprints encapsulate the specialized skills, knowledge, and expertise of infrastructure experts and are centrally stored, allowing DevOps teams to manage and share them in a controlled manner easily.

As a result, DevOps enables developers, testers, and engineers to do their jobs faster, more efficiently, and more securely.

ENHANCED SECURITY

Simloud enhances the security associated with accessing and managing infrastructure, making it easy for DevOps to enforce policy, compliance, and governance throughout the entire lifecycle of an infrastructure environment.

- An automated and custom tagging feature establishes a foundation for resource utilization metrics and cost accountability, capturing details on how infrastructure is being used and by whom with associated costs.
- Simloud's automated infrastructure teardown capability prevents cloud resources from being abandoned and helps eliminate cost overruns as cloud infrastructure often needs to be dynamic and short-lived.
- With Simloud, DevOps can become consultative service providers to all stakeholders in their organizations, delivering complete infrastructure environments as a service eliminating provisioning bottlenecks. These environments include the foundational infrastructure, application configurations, and contextual elements needed by applications to run successfully while fostering transparency and optimized governance.

It is a proactive approach to building and providing infrastructure services that optimizes the value of an organization's infrastructure investment.

Simloud increases DevOps velocity with environments provisioned on-demand

Software drives business innovation today. As companies seek greater agility, they're adopting increasingly complex software stacks. It makes software architectures evolve rapidly and become services-based and dynamic with their components running in the cloud, containers, micro-services, and highly distributed infrastructure.

That's why DevOps teams need an efficient way to provision these complex application environments while accelerating software release velocity. At the same time, it has to be scalable, frictionless, controllable, and visible. This is what Simloud offers.

Simloud's Environments-as-a-Solution platform provisions application environments on-demand consistently and at scale, regardless of technology. Simloud defines infrastructure in modular blueprints including all the elements an application needs throughout its life cycle.



PROVISIONING ENVIRONMENTS ON-DEMAND

Modular Blueprint Modeling

Simloud's self-service functionality incorporates modular blueprints defining the complete environment that an application needs to run successfully. It includes the infrastructure, data, network topology, cloud services, and software services. Administrators control user access to blueprint elements, resources, and accounts, while environment setup and teardown are automated in a repeatable and reliable manner.

Simloud leverages existing Infrastructure-as-Code (IaC) and application configurations from tools already in use by development and delivery teams. With Simloud, DevOps teams can respond to business requirements to deliver software faster and with greater agility, creating a competitive advantage and fueling innovation.

Self-service

Simloud gives developers full self-service, on-demand access to the environments they need, eliminating provisioning bottlenecks. Role-Based Access Controls (RBAC) provide access only to those resources that the DevOps team defines as appropriate. Simloud provides access to cloud infrastructure and application environments through the Simloud GUI, users' preferred CI/CD, RestAPI, and/or CLI.

Seamless CI/CD Integration

Simloud supports full integration with your Git repositories for infrastructure definition and for CI/CD ops. Simloud enables automatic Jenkins master/slave configuration, thus supporting scale from day one with pre-configured CI/CD pipes, available for both Serverless and/or Kubernetes services. Simloud also natively integrates into an organization's existing DevOps ecosystem and CI/CD pipeline tools to support continuous build and deploy systems throughout the entire DevOps process.

Frictionless Governance and Accountability

With Simloud, users control and manage access to blueprints to ensure compliance and prevent uncontrolled infrastructure and cloud usage. Auto-tagging eliminates uncertainty and avoids cloud account overspending.

Users can view cloud cost per environment in the reach dashboard. These insights help identify opportunities for resource consolidation and build accountability for resource usage and cloud costs.

With its robust controls and self-service access to complete application environments, Simloud accelerates developer velocity and software innovation, unleashing the productivity of DevOps teams to address and manage complexity across the entire software life cycle. The platform helps digital enterprises succeed and achieve new business results powered by modern software.

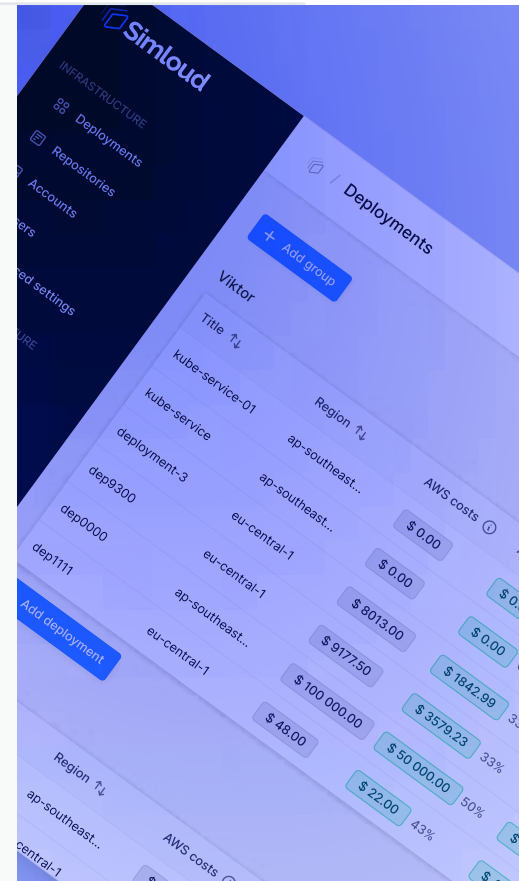


Simloud Low-Code to No-Code Automatic Internal Developer Platforms and Ops (AIDPOs)

While self-built Internal Developer Platforms (IDPs) have been around in elite teams for about 5 years, they became mainstream in 2021. Now, they are also used by global enterprises like Spotify, Airbnb, and Zalando.

Simloud low-code to no-code approach to Internal Developer Platform and Operation (IDPO) is a layer on top of the tech and tooling that's already in place. Simloud is configured and set up by DevOps teams and then used by developers as self-service. DevOps teams specify what resources are used in specific environments or at what request. They also set baseline templates for application configurations and govern permissions. This helps them to automate recurring tasks, such as environment and resources spinning up, and makes their setup easier to maintain by enforcing standards.

Developer teams gain autonomy by changing configurations, deploying, spinning up fully provisioned environments, and rollback.



How is Simloud used by DevOps or Platform teams?

The DevOps team primarily configures Simloud and concentrates on infrastructure, service level agreements (SLAs), and workflow optimization to abstract away any recurring or repetitive tasks, such as spinning up resources or environments for developers. The team also sets baseline templates for configuration and avoids unstructured scripting to prevent excessive maintenance time.

How is Simloud used by application developers?

Simloud is integrated into existing workflows which will usually remain a git-push deploy workflow but with further automation. The entire deployment process is now at the disposal of the developer. They can request resources, spin up fully provisioned environments, roll back, deploy and set deployment automation ruling autonomously.



CODE

Code where you always code



MERGE

Git-push as usual



RUN

Run on your Internal Developer Platform

A modern developer needs three panes of glass: the IDE to code, git to merge and Simloud and an IDP to ship.

The five core components of Simloud:

Two features are exclusively used by the DevOps or Platform team:

- [Infrastructure Orchestration](#) to integrate with your existing and future infrastructure.
- [Role Based Access Control \(RBAC\)](#) to manage who can do what in a scalable way.

The feature used by the DevOps team:

- [Application Configuration Management](#) is used to set baseline-templates but also used in day-to-day activity by the application development team.

Developers use the following functionalities:

- [Deployment Management](#) to implement a Continuous Delivery or Continuous Deployment (CD) approach.
- [Environment Management](#) enables developers to create new environments whenever needed.

APPLICATION CONFIGURATION MANAGEMENT

Application Configuration Management can be a real nightmare. While handling application code is already a well-standardized process, managing application configuration is not. Whatever your approach, you are left with a number of config files (typically in YAML format) that you and your teams/colleagues need to maintain. It might be a simple task as long as there are no substantial changes and as long as everything works. But the tasks can quickly get out of control if, for example, the developer responsible for the setup left, or the setup stopped functioning, or it needs to be extended.

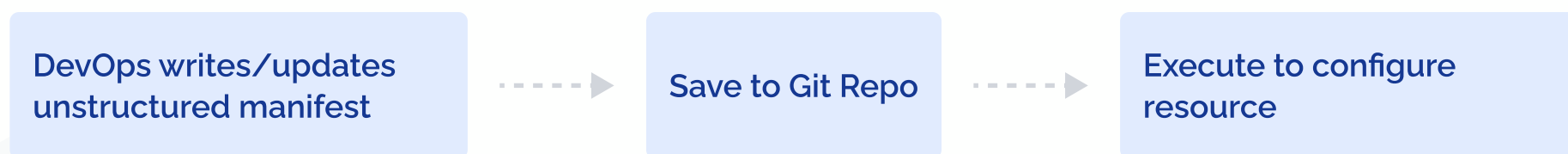
Simloud provides a standardized approach to Application Configuration Management that has a lot of similarities with using Git for managing source code. This section describes the general approach to Application Configuration Management with Simloud.

Typical challenges *without* an Internal Developer Platform

- 1 Configuration is often saved in script or YAML files. Maintaining these files can be hard even if you are applying an approach such as GitOps to them.
- 2 Versioning configuration can be hard, especially since configuration very often needs to be altered depending on the environment you want to use it with. How do you distinguish between environment-specific and non-environment-specific elements in your application configuration?
- 3 Most current setups to manage application configuration do not allow for self-service from a developer. Thus, setting up a new environment for a feature branch or for a manual QA test needs to involve an expert from the DevOps team (which is typically not what your DevOps team should focus on).

Simloud Approach

- 1 **Scope:** Simloud manages both, resources that are living within your containerized application platform (e.g., Kubernetes) as well as resources that are running outside of it (e.g., databases, file storage). This is important since you typically need both kinds of resources within any given modern application.
- 2 **Versioning:** Simloud versions a baseline configuration that is changed for any specific application and environment, and are externalized via Simloud user interface (UI), command-line interface (CLI), and API to accommodate different preferences within your developer teams.
- 3 **Portability:** Simloud allows you to store standard manifests for each deployment in a Git-based repository.
- 4 **Secrets Management:** Managing secrets comes with a lot of challenges, especially in the everyday life of developer teams. There are still too many secrets checked into (maybe even publicly available) repositories; often accidentally. Simloud supports your DevOps team as well as your developers in managing secrets in a convenient and secure way stored in Hashicorp Vault and pushed to your service on your CD process.



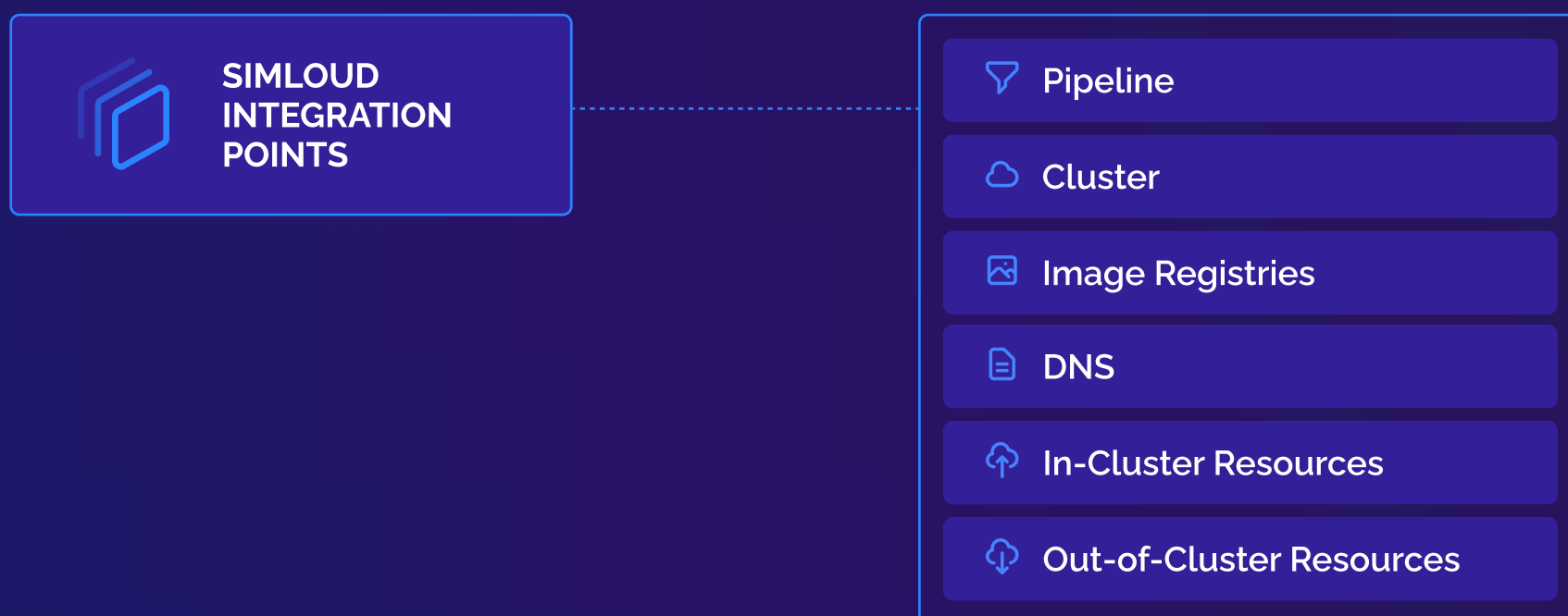
Application Configuration Management **without** an Internal Developer Platform

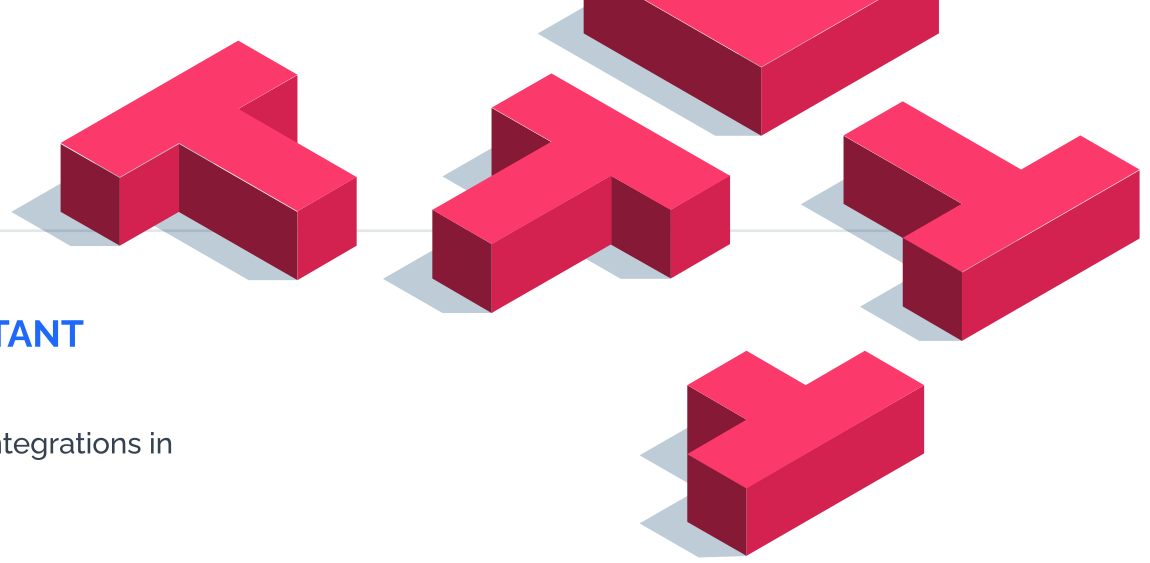


Application Configuration Management **with** an Internal Developer Platform

INFRASTRUCTURE ORCHESTRATION

Simloud provision for you end-to-end VPC with a seamlessly integrated infrastructure and it orchestrates your existing and future infrastructure. Simloud would either create your CI/CD pipeline allowing you to configure or integrate with your existing CI pipelines on the one side and your cloud infrastructure (e.g., Kubernetes clusters, databases, file storage) on the other side. This approach differs from a PaaS (Platform-as-a-Service) that typically includes all the infrastructure (often based on proprietary technology stacks).





OVERVIEW OF SIMLOUD MOST IMPORTANT INTEGRATION POINTS

Important: Simloud allows you to write your own integrations in case you need to do so.

CI pipelines

Simloud ensures a seamless integration with your CI pipelines. Setting up and configuring Continuous Integration (CI) pipelines can be time-consuming. Simloud is connected with your CI pipelines. It knows when a new image is available so that it triggers the next required step in your Continuous Delivery or even Continuous Deployment (CD) process (configurable).

Kubernetes Clusters

Kubernetes clusters are an essential element to run your containerized setup. Simloud integrates with your existing clusters to run deployments of applications and environments.

Image registries

Simloud creates and manages the images used in your applications. Depending on your specific technical setup, Simloud can provide a built-in image registry (CI pipeline integration will push new images to this registry) or use the fact that your image registry is already connected to your clusters (in this case, it will trigger the deployment process directly via already connected cluster API). If you go for a streamlined user experience, a built-in image registry might be more relevant to you. If your image registry is well established and includes a lot of security and vulnerability scans already, you might want to stick with it.

DNS

Enabling developers to create new environments whenever needed is a critical aspect of Simloud. Providing environments on demand allows for issuing new subdomains for the specific environment. Simloud integrates with your DNS provider to enable this functionality.

Other resources

There are many more resources relevant for your specific applications. These resources run in your cluster (in-cluster resources, such as messaging queues or caching databases) or outside your cluster (out-of-cluster resources, such as databases or persistent file storage). Simloud allows simple integration of both.

IaC

Simloud integrates well with an existing Infrastructure-as-Code (IaC) setup. It directly enables IaC as a native feature and supports it via integrations. The latter is handy when you want to leverage existing approaches, such as Terraform, to manage your IaC.

ENVIRONMENT MANAGEMENT

Simloud allows developers to self-serve new environments on-demand. It removes a lot of bottlenecks and enables faster delivery. Each new environment is provisioned as defined by the DevOps team.

From a developer perspective, Environment Management is one of the most practical components of Simloud. Usually, creating and configuring a new environment involves waiting for someone else (most likely the DevOps team), which is time and money-consuming. Simloud eliminates the required manual steps and enables self-service for the developer (or other people in your team) to spin up a new environment whenever needed.

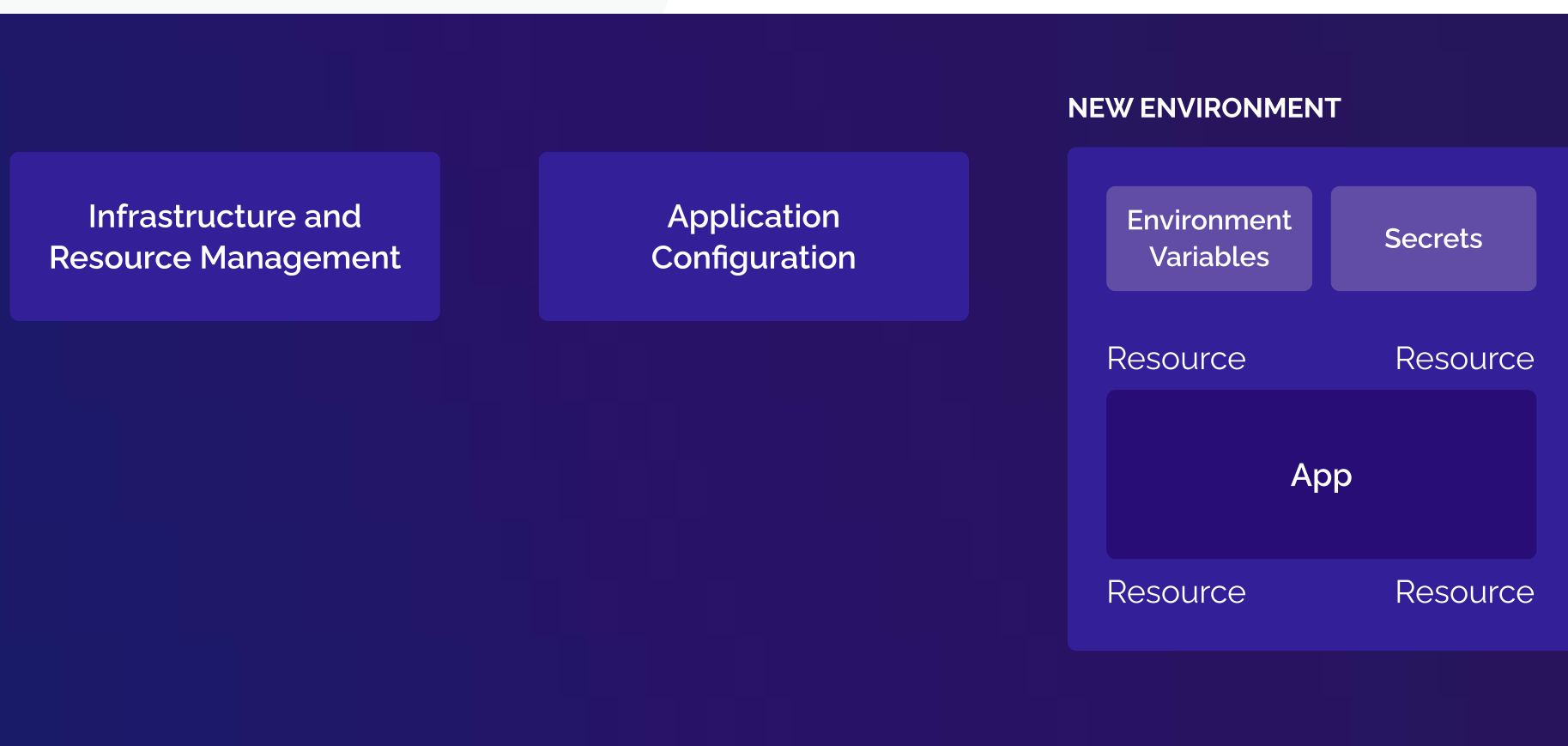
Challenges of manual setup

There are many challenges when it comes to Environment Management:

- Setting up a new environment requires reaching out to another team (e.g., the DevOps or platform team). This may take hours if everything is well organized but might also take days or even weeks if the team is working on other priorities.
- Complex environment setup may not always have a switch-off option for the non-used environments. This means that you'll need to keep them just in case and pay for them even if they do no work.
- Not being able to create new environments when needed can significantly impact your speed of delivery. This is a common developer's issue. For example, when shared environments are blocked for days or even weeks because one team needs to test a hotfix that needs to go to production asap or because QA feature testing takes too much time. When these environments are blocked, other teams cannot test their own new services or features, and, as a result, the whole project faces bottlenecks.

Another problem is that infrastructure, application configuration, and the real environment are often managed in silos. It's not practical from the organizational and technical point of view: on the one end, it requires constant maintenance of integration points between the siloed solutions, and on the other — strong developer skills with multi-cloud disciplines (e.g. Kubernetes, security, CI/CD, etc.). Naturally, it evokes complications.

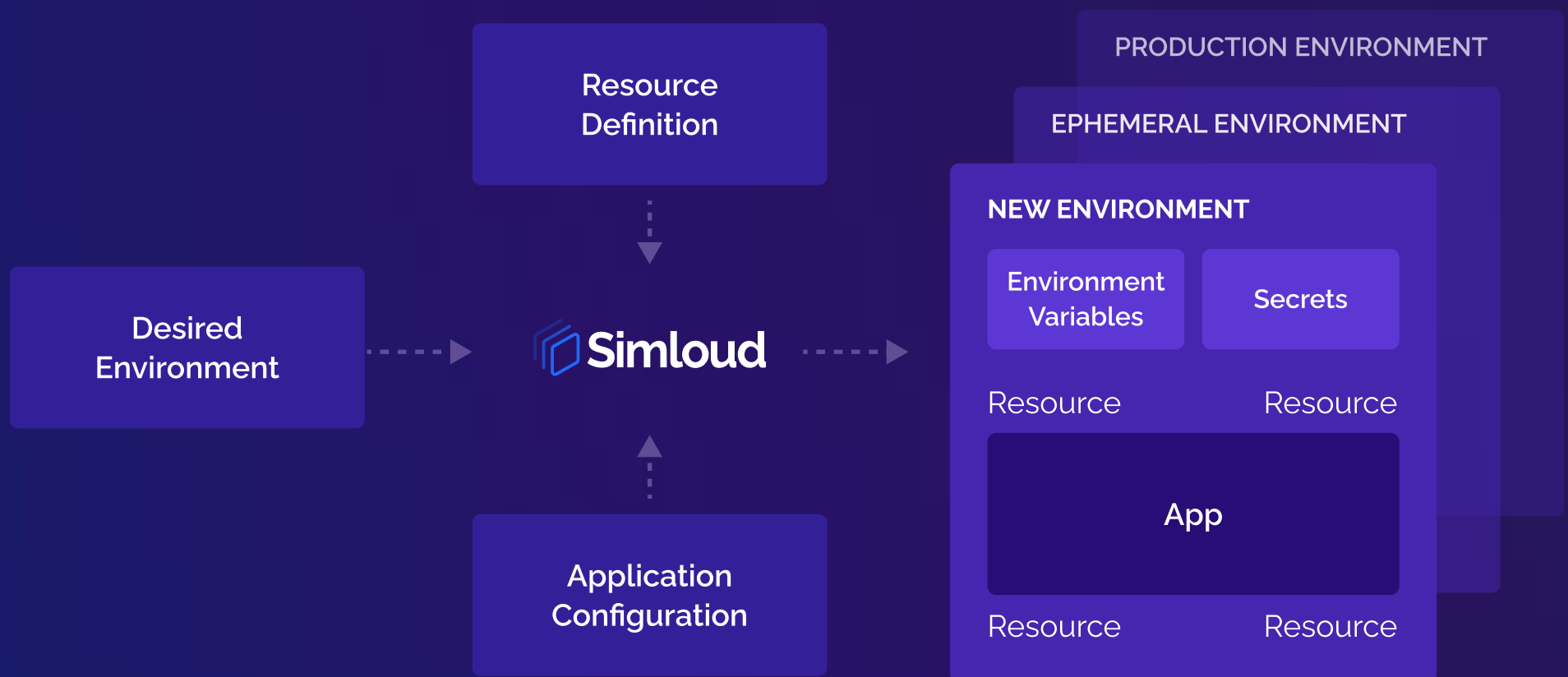
Here's what the siloed approach looks like:



Infrastructure, application configurations, and environments are typically managed in silos.

Simloud solutions

Simloud connects infrastructure, application configurations, and the management of environments to boost the developer experience. Here's what the Simloud solution looks like.



Simloud ties together infrastructure, application configurations, and environments to boost the developer experience.

There are a number of different elements of how Simloud supports Environment Management:

- **Self-service.** Simloud enables developers to create new environments when needed. New environments are created within the context of an application typically by cloning an existing deployment to a new environment. The developer can alter the newly created environment however needed (e.g., to test a new service or feature branch). All of this should happen based on the infrastructure provided and maintained by the DevOps team. Simloud provides different ways to create new environments via a user interface (UI), a command-line interface (CLI), or an API - which is typically the best solution for fully-automated environment creation (e.g., for automated end-to-end tests). Simloud also supports automated teardown or pausing of environments if they are not needed to avoid unnecessary costs.
- **Ability to define environment types.** With Simloud, the DevOps team can create new environments with reasonable infrastructure requirements (e.g., small machines for development environments and a powerful setup for production or a load test environment).

DEPLOYMENT MANAGEMENT

Deployment Management within Simloud enables teams to move to a Continuous Deployment (CD) process. It also provides a clear record of each deployment ever made which is great for audits and similar processes.

Simloud supports teams in establishing Continuous Delivery or even Continuous Deployment (CD) processes. Deployment Management plays a central role in this. However, Deployment Management is so much more than just automated deployments. This section below explains the typical elements of Deployment Management within Simloud.

The ideal development process with Simloud

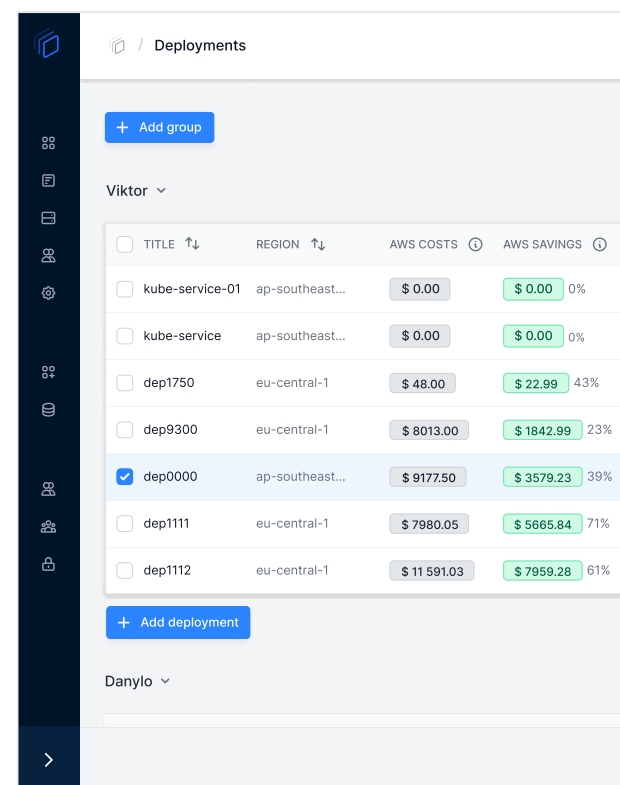
In an ideal setup, developers can fully focus on writing code and testing it in action. Everything else should be automated as much as possible. Simloud allows teams to set up this automation through the following process.

- **Git push:** The developer pushes new code to the version control system. The new code is then picked up by the CI pipelines and built in an image. The CI process typically also includes a number of automated tests to ensure that the code is well written. At the end of the CI process, Simloud is notified that a new image is available.
- **Automated deployment:** The newly built image is deployed to a specific environment. Typically, this is the first step in development. A set of rules defined in Simloud orchestrates to which environment the new image is deployed. The entire deployment process should be fully automated without any interaction from the developer.
- **Triggered next steps:** Not only does Simloud automate the deployment process itself but also supports defining the next steps (typically via webhooks) after each deployment. In the case of a successful deployment, these next steps could be (a) sending a message about the successful deployment to the developer (e.g., via communication services such as Slack or MS Teams) and/or (b) triggering a fully automated end-to-end test suite for the new deployment. Simloud also supports more complex processes including numerous steps and checkpoints to enable Continuous Deployment.

Additional aspects of Deployment Management

While streamlining developer experience is a key aspect of Deployment Management, there are many others:

- **Debugging support:** What happens if the deployment as described in the last section fails? Debugging failed deployments can be a time-consuming task when using multiple different systems with different logins and user interfaces/log files. Simloud provides a consolidated view on the most important debugging information (e.g., deployment logs, container logs) for any current or even past deployment. This centralized information is a great starting point for debugging a problem and can save a lot of time and headaches.
- **Versioning:** Simloud acts like a central memory for all deployments ever made and stores all required information to repeat a specific deployment. It significantly simplifies audit processes and enables Git-like actions such as diffing two deployments or creating patches for deployments. It can also answer questions like: "Which version of any given service is running where?", "In which environments was a certain version of a service deployed and tested before being released to production?", and others.



TITLE	REGION	AWS COSTS	AWS SAVINGS
kube-service-01	ap-southeast...	\$ 0.00	\$ 0.00 0%
kube-service	ap-southeast...	\$ 0.00	\$ 0.00 0%
dep1750	eu-central-1	\$ 48.00	\$ 22.99 43%
dep9300	eu-central-1	\$ 8013.00	\$ 1842.99 23%
dep0000	ap-southeast...	\$ 9177.50	\$ 3579.23 39%
dep1111	eu-central-1	\$ 7980.05	\$ 5665.84 71%
dep1112	eu-central-1	\$ 11 591.03	\$ 7959.28 61%

ROLE-BASED ACCESS CONTROL

With Simloud, the DevOps team manages access on a granular level. This limits access to production to a small number of trusted people while allowing every engineer to create new development environments as needed.

Different people across an IT organization can use Simloud to get their daily work done more efficiently, especially if it has something to do with managing access and permissions. The DevOps team can use Simloud to set up the infrastructure, the product manager can test a new feature in a realistic environment, and the engineer can release new features to production.

This section provides an overview of the most important functionality of Simloud regarding access control.

Enterprise-level granular control

There are many different ways of managing access rights and permissions in a software system. However, the most popular approach is Role-Based Access Control (RBAC), especially in an enterprise context. According to RBAC, the roles are created for individual job functions. Permissions are then assigned to these roles. Everybody in an organization receives one or more roles. This approach is much more scalable than assigning permissions to individual users. Simloud offers Role-Based Access Control to enable scaling.

Ideally, roles should be split into organizational roles and application-specific roles:

- **Member:** The role of a developer in a team. Members can typically access the applications they are working on.
- **Machine:** This role can have different names. It's often a role that can be used for infrastructure integrations (e.g., for a CI pipeline pushing images into Simloud). It has very limited rights.
- **Manager:** The role of an engineering manager. Managers can typically invite users and manage the applications the team is responsible for. The scope might vary between different implementations.
- **Admin:** The role of the DevOps team or lead. Admins have full access to the entire functionality of Simloud.

Large organizations often require roles to cover specific requirements.

Typical application-specific or even environment-specific roles are:

- **Viewer:** A read-only role for an application or an environment.
- **Contributor:** A role that can update the configuration for an application and create new environments. This role is typically used in combination with the Member role mentioned above.
- **Owner:** A role for the owner of a specific application with full access. Typically, only owners can delete an application.

The ability to manage access on an environment level is important in Simloud. This allows your Admins to limit full access to a production environment to few people while enabling other people to create and update development environments. Simloud will soon support an automated mapping to the company's directory information service (e.g., LDAP) to cater to the needs of organizations based on their size.

When do you need Simloud?

When is Simloud useless?

- You have a single monolithic application.
- You have one application with a simple, single-cloud infrastructure.

When is Simloud useful?

- You have or you plan to adopt a microservice architecture.
- You have a standing team of more than 3 developers with a dedicated DevOps engineer or you are planning to scale to this size.
- You have a DevOps team (or person) that cannot properly focus on service level agreements or workflows because of repetitive work.
- You have a small team and not everyone feels comfortable with deployments, scripting, and infrastructure.
- Your developers are blocked in their work by dependencies on other colleagues.
- You have to go multi-cloud.

