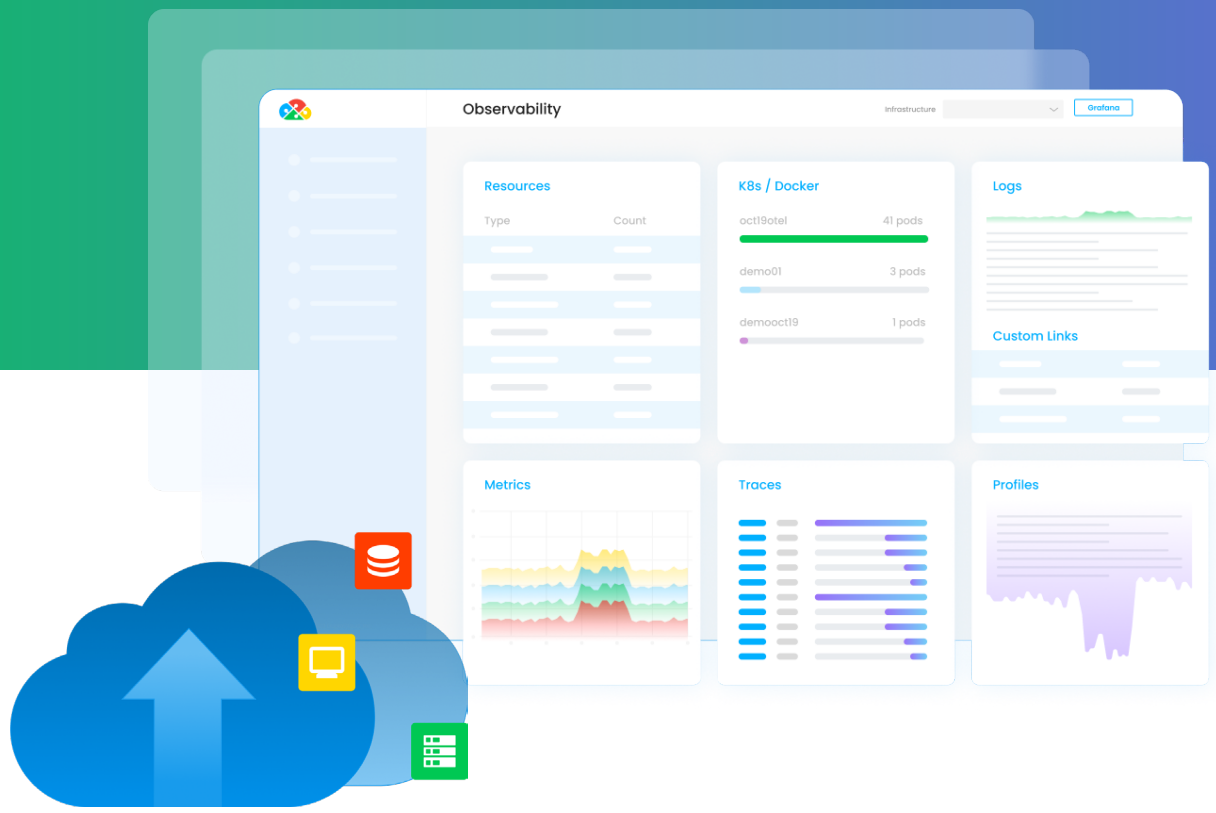
DuploCloud

WHITEPAPER

# Platform Engineering

## The Rise of the Internal Developer Platform for Cloud Deployment and Operations



# Contents

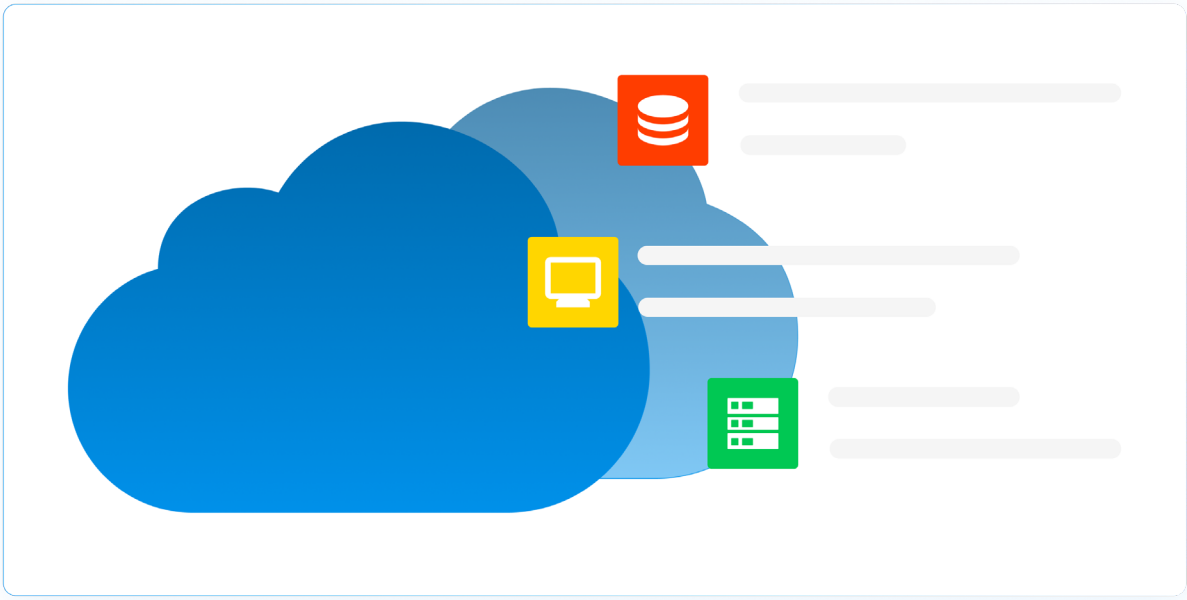
# Introduction

**The Rise of the Internal Developer Platform for Cloud Deployment and Operations**

The adoption of Service Oriented Architecture (SOA) at AWS and Azure gave birth to the original DevOps culture where Developers would own the end-to-end lifecycle of an application from coding and running deployments to maintaining uptime of the application. Unfortunately, today's DevOps is not about Developers owning operations, but rather operators building automation for their own operational efficiencies.

Developer self-service with respect to cloud infrastructure is quite scarce in most organizations. Developers raise support tickets to DevSecOps and wait days for them to be fulfilled. In organizations where developers are allowed unfettered access, the security of the cloud infrastructure is in disarray: open ports, unchanged passwords, untracked keys, unencrypted disks, etc. Many organizations are trying to address this problem by creating Platform Engineering teams. The lofty goal: build an Internal Developer Platform (IDP) to improve engineering productivity through developer self-service, with security "guard rails". This dedicated and experienced team of engineers who have been assigned the task will likely spend several months to years building and maintaining their in-house IDP.

In this whitepaper, we describe how the DuploCloud DevOps Automation Platform can be your out-of-the-box IDP. Many organizations have also built a layer of customization on top of DuploCloud to add workflows not supported natively, saving millions of dollars and years of effort.



## Infrastructure-as-Code alone is not an IDP

Any modern-day application consists of many independent pieces, often called microservices. These include both cloud provider services like S3, SQS, Kafka, Elasticsearch, etc. as well as application components owned by the organization and deployed as Docker containers in Kubernetes. Cloud providers support hundreds of services for applications to use. While this has obvious advantages of scale, availability, and agility, it is extremely hard to manage — too many moving pieces, access controls, thousands of nuanced of infrastructure configurations, hundreds of compliance controls, and more. Infrastructure-as-Code (IaC) is a scripting language that is optimized for building and operating these configurations. But there are several limitations with IaC in its current form:

## DevOps is a very difficult skill

DevOps demands a single individual to be proficient in operations and security, as well as programming (i.e., IaC or Infrastructure-as-Code). These have traditionally been three independent job profiles. Developers are not operators. Operators' programming skills are limited to basic scripting and most operators don't have a good grasp of compliance standards. There are ready-made libraries or modules for some standard functions, but nevertheless, an engineer without a sound operations background cannot build and operate IaC.

## IaC cannot enforce compliance by itself

Being a scripting tool that requires attended execution, the scope of the system is limited to the time when the user executes it. There are many scenarios where the infrastructure may deviate from the desired state, which includes users making changes directly in the cloud. So, one needs to build out-of-band systems to monitor these that would alert a user to take corrective action manually. Compare this with intent-based Configuration Management systems like Kubernetes, AWS, Azure, etc. where once the intent is configured in the platform, the system drives the underlying infrastructure to the goal state, detects drifts, and performs remediation.

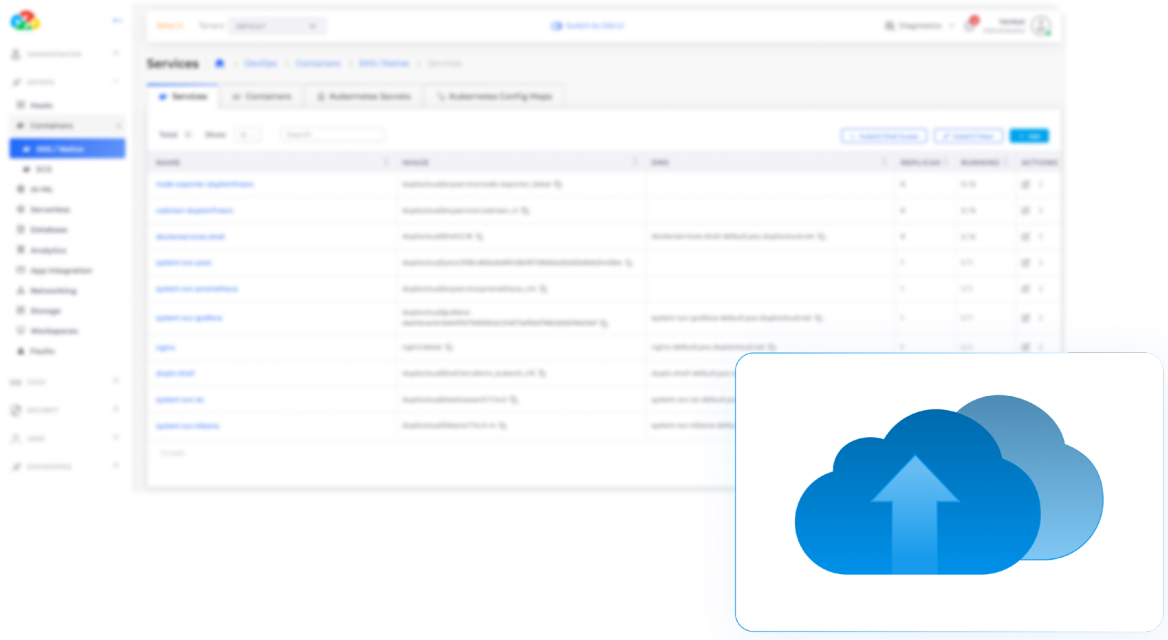
## Lack of ability to track Intent

None of the platforms (Azure, AWS, Kubernetes, etc.) are built on top of scripting tools. They are all written with higher-level programming languages. IaC is a scripting tool that executes instructions serially and is meant to be attended by a human. A self-service cloud automation solution requires an intent-based platform where you define a higher-level specification and the platform asynchronously applies the configuration to the cloud provider by coordinating various dependencies in a state machine. You cannot build a self-service cloud management platform using Terraform.

## IaC does not provide a UI, or RBAC, and does not manage Access Control

For ongoing operations and debuggability, multiple users need scoped access to cloud components. Role-based access, JIT access control with the principle of least privilege, and integration of operational elements need to be built. They are not in the scope of IaC.

It is unrealistic to expect that developers would own the end-to-end lifecycle using only IaC automation and achieve developer self-service, from coding and running deployments to maintaining the uptime of the application. An IDP that assumes these tasks, while providing a system that is self-service, with minimal requirements for operational and security experience, becomes essential.



## Desired Goals and KPIs for an IDP

As with all software and projects, it is important to have clear goals and KPIs. In the case of infrastructure automation, goals and KPIs are critical for defining the broad spectrum of automation. Following are the key goals that we set while building the DuploCloud DevOps Automation Platform. We show the KPIs we have tracked towards those goals:

## Reduction in manual labor and Cost Savings

The bottom line to success in cloud automation is reducing the level of human involvement in daily measurements. The best way to measure this is by counting the number of DevOps engineers an organization must employ, proportional to the size of their cloud workload, measured in terms of either virtual machines or cloud services. **Figure 1** shows the quantification of this metric. In most organizations, SecOps is a dedicated job profile. If the IDP is built right, then compliance and security do not require a separate head count.

As detailed in the blog Are You Spending Too Much on DevOps?, 80% of the DevOps cost is manual labor, while 20% is tools. Using DuploCloud, required resources are reduced by an order of magnitude and the efficiencies of reduction in manual labor reflect directly in cost savings.

| Infrastructure Size | Inhouse DevOps Engineers |
|---|---|
| Less than 50 VMs and 10 Micro-services | 0 - 1 |
| 50-200 VMs and 30-50 Services | 1 - 2 |
| > 200 VMs and 100+ Services | 2 Engineers + (1 Engineer for every 200 VMs) + 1 SecOps Engineer |

Figure 1: KPI for Reduction of Human Labor and Operational cost

# Comprehensive Automation Platform

An IDP should automate most of the low-level tasks and expect users to only specify high-level intent. This ensures that developers can get things done without knowing low-level details. While DevOps automation is a broad spectrum, you should strive to automate 95% or more of your functionality out-of-the-box, in the platform. The KPIs for this goal are the number of cloud automation functions, cloud provider services, and third-party tools that can be deployed using the platform. **Figure 2** shows the representative services that DuploCloud's platform supports, and new services are added on a monthly release cadence. User-requested services typically take 1-2 weeks. Once added to the platform, these services are available to all users.

| | | | |
|---|---|---|---|
| **CICD** | | | |
| Integrations with Jenkins, GitHub Actions, Bitbucket, Gitlab, CircleCI, Azure DevOps | DAST | SAST | Self-Hosted Runner Management |
| **Observability and Diagnostics** | | | |
| Central Logging with Open Search | Metrics with Prometheus, Grafana, Azure Mon, Cloud Watch | Alerting with Pager duty, Sentry and New relic | Audit Trails |
| **Application Provisioning** | | | |
| **Containers** | **Big Data** | **Serverless** | **AI/ML** |
| Kubernetes; ECS; Azure Webapp; GKE Auto Pilot | Airflow On K8S (MWAA); Spark, EMR; Glue; Datapipe line; | Lambda; Azure Functions; GCP Cloud Functions; AWS Batch; CloudFront | Sage Maker; Kubeflow; Azure ML Studio |
| **Cloud Platform Services (AWS, Azure and GCP)** | | | |
| **Managed Services** | **Access Control** | **Connectivity** | **Configs and Secrets** |
| 200+ cloud PaaS services like Managed Databases, Redis, Managed Kafka, Message queues, SNS, Service bus, S3, | Single-sign on; Just-in-time access; Local Development; Kubectl, App Shell and VM SSH | Load Balancers, Ingress, DNS, WAF, Security Groups | Secrets Manager, SSM, K8S config map, secrets, Azure Key Vault |
| **Data Protection and Backup** | **Encryption** | **Cost Management** | **High Availability** |
| Snapshots, Azure Backups, Log Analytics, Database and Open Search backups | KMS, Certificates | Per service and Per Tenant cost views Resource Tagging; Resource Quotas; Billing Alerts | VM Auto-scaling; Kubernetes Cluster and Pod Auto-scaler; Availability Zones, Multi-Region Deployments |
| **Networking and Guard Rails** | | | |
| VNET, VPC | Subnets and Routing | VPN and Peering | CloudTrail |

Figure 2: Representative Services supported by DuploCloud as KPI for Comprehensiveness of the Platform.

## Developer Self-service

While this is an important goal and KPI for an IDP, it is also difficult to quantify, as developer skill levels vary widely. We have chosen to quantify this goal using the metrics shown in **Figure 3**. You can see 50,000 infrastructure changes are enabled across 75 organizations, with an overwhelming number of users being developers. Across our user base, there are only 35 DevOps people for 800 developers, which is a very low number for this scale of infrastructure.

| | | | |
|---|---|---|---|
| Customers | 75 | VMs | 2,200 |
| Developers | 800 | Containers | 7000 |
| DevOps | 15 | Unique Cloud Services | 200 |
| Cloud Providers | 4 | Avg Infra changes/mo | 50,000 |
| Cloud Spend under management | $8M/Yr | Compliance certifications/yr | 45 |
| 170% YOY growth in User base and Infrastructure under management | | | |

Figure 3: Developer Self-Service KPIs. All Numbers cumulative across clients

## Time to Compliance

Compliance to regulatory standards has become a table stake requirement to operate cloud infrastructure. Security and compliance cannot be an afterthought for an IDP. An important metric for an IDP should be time to compliance, as shown in **Figure 4**, and if the organization is operating in multiple verticals, then all of those would need to be supported.

For the majority of our customers, their primary motivation for adopting DuploCloud was to achieve regulatory compliance for their cloud infrastructure. DuploCloud's automation approach is inherently secure and compliant as the platform bakes in compliance controls during infrastructure provisioning.

| | |
|---|---|
| Standards Supported out-of-box | 10+ |
| Avg. Time to Implement | 2-4 Weeks |
| Number of unique customers Certified/yr | 45 |
| Biggest Infrastructure Certified | 400 VMs, 1,000 Containers |
| Avg Audits per month across the customer base | 4 |

Figure 4: Compliance KPIs

## Secure by Design

DuploCloud's platform controls the end-to-end configuration stack, covering more than 80% of controls in various security standards, as shown in **Figure 5**.

| Standard | Controls Implemented | Detailed Documentation |
|---|---|---|
| **SOC2** | 80 | https://duplocloud.com/white-papers/soc-2-compliance-with-duplocloud/ |
| **HIPAA** | 29 | https://duplocloud.com/white-papers/pci-and-hipaa-compliance-with-duplocloud/ |
| **PCI** | 79 | https://duplocloud.com/solutions/security-and-compliance/pci-dss/ |
| **ISO** | 50+ | |
| **HITRUST** | 75+ | |
| **NIST** | 200+ | |

Figure 5: Secure by Design KPIs

## Design and Architecture

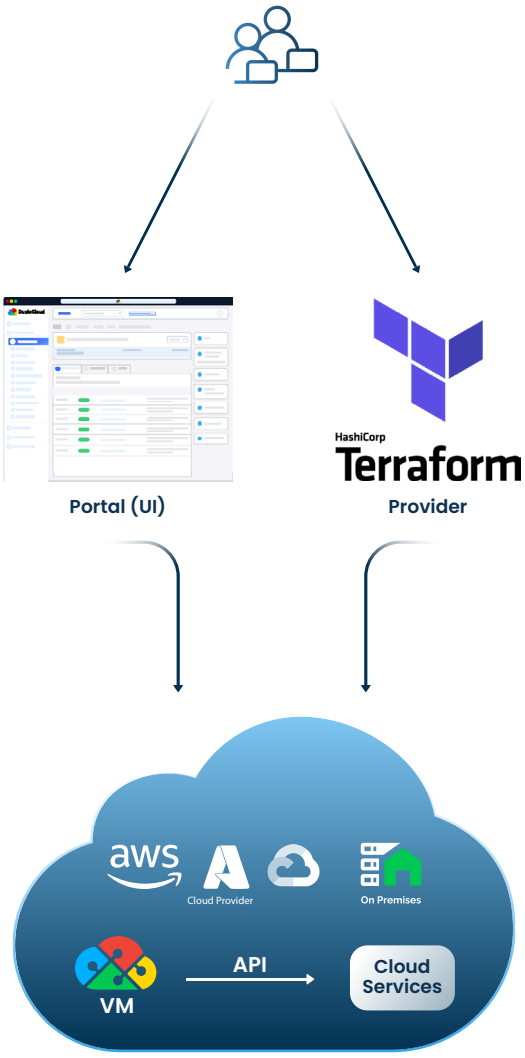
The founding team at DuploCloud were among the original inventors of the Public Cloud working for Azure and AWS back in 2008, having built the platform where millions of workloads are deployed across the globe, managed with just a handful of operators. The design of DuploCloud comes from their learnings and experience in this hyper-scale environment. There are 6 key elements to the DuploCloud design:

## Self-Hosted and Single Tenant

The DuploCloud platform is a self-hosted solution that is deployed within the customer's cloud account. It inherits permissions from the Instance Profile/Managed Identity of the VM and manages the environment through cloud provider APIs. With the customer's permission, DuploCloud provides a fully managed service to maintain uptime, updates, and ongoing support. In the case of AWS, each account has a DuploCloud VM and a unique endpoint in alignment with the IAM architecture that is tied to an account. In the case of Azure, a single DuploCloud VM maps to an AD and can manage multiple subscriptions.

# No-code / Low-code UX

DuploCloud gives the option to use both a purely no-code UI or a low-code Terraform provider (for those who prefer IaC). DuploCloud's Terraform Provider is similar to an SDK in Terraform that allows the user to configure cloud infrastructure using DuploCloud constructs, rather than lower-level cloud provider constructs. This allows the user the benefits of Infrastructure-as-Code, while significantly reducing the amount of code that needs to be written. The DuploCloud Terraform Provider simply calls DuploCloud APIs. Our DevOps White Paper provides detailed examples.

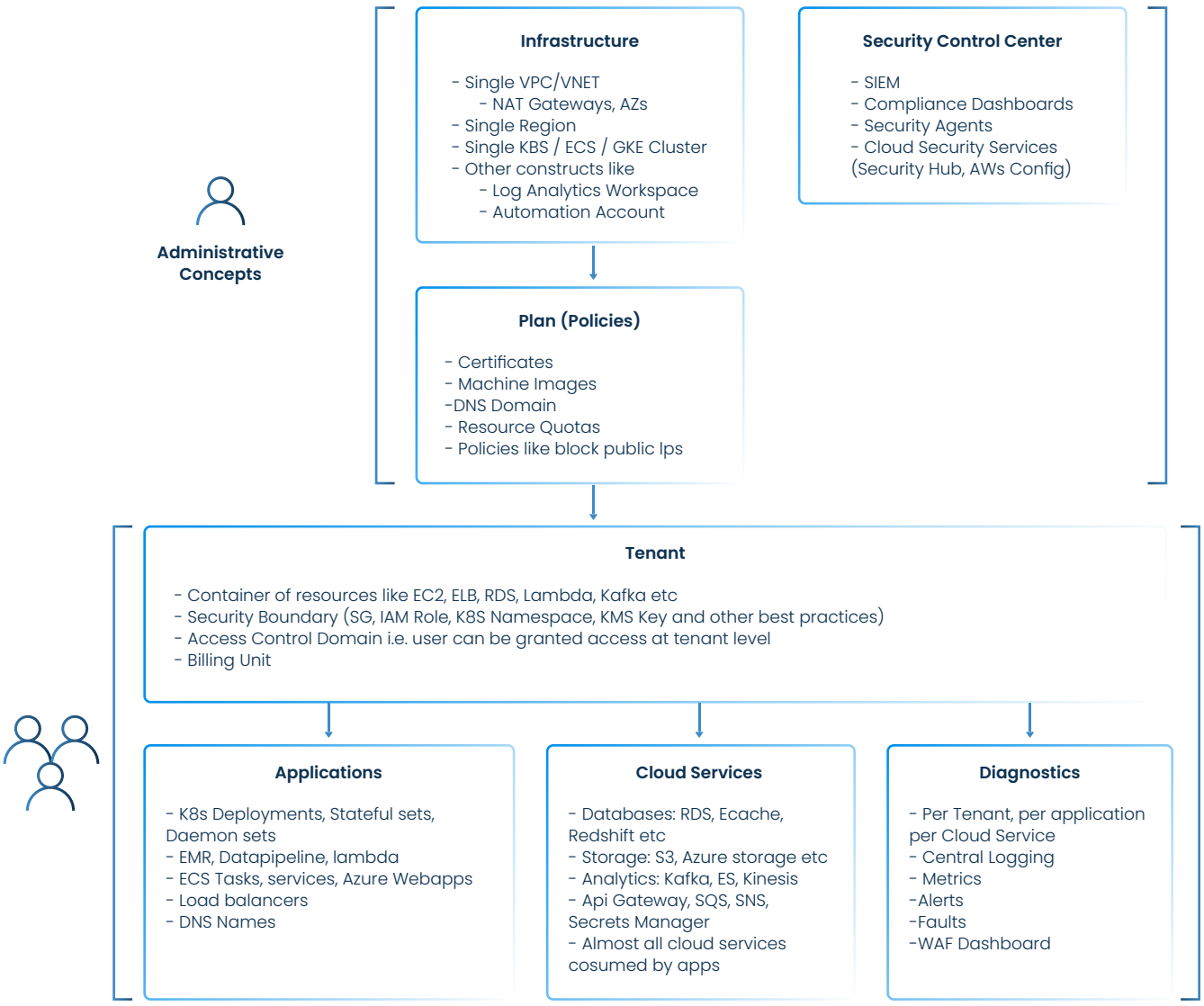



It is important to note that Terraform is a layer on top of DuploCloud and DuploCloud does not generate Terraform underneath to provision the cloud provider, rather DuploCloud's provisioning is via native cloud APIs.

# Application-Focused Constructs / Policy Model

The greatest capability of the DuploCloud platform is the application-centric abstraction created on top of the cloud provider, which enables the user to deploy and operate their applications without knowledge of lower-level DevOps nuances. Further, unlike a PaaS such as Heroku, the platform does not get in the way of users consuming cloud services directly from the cloud provider, meaning that a user can directly operate on constructs like S3, DynamoDB, Lambda functions, etc, resulting in greater scale and unlimited flexibility.

Some concepts relating to security (DevSecOps) are hidden from the end user (IAM roles, KMS keys, etc.). However, even these are configurable for the operator. Since this is a self-hosted platform running in the customer's own cloud account, the platform works in tandem with direct changes on the cloud account by an administrator. This is explained with examples in our DevOps Automation Whitepaper.



**While there are many concepts in the policy model, the key components are:**

**Infrastructure**
Each Infrastructure is a unique VNET, in a region with an AKS cluster and Log Analytics workspace, among other constructs.

**Tenant**
A Tenant is the most fundamental construct in DuploCloud. It is a project or a workspace and a child of the infrastructure. While Infrastructure is a VNET level isolation, Tenant is the next level of isolation, implemented by segregating Tenants using Security Groups (SGs), Managed Identity, Kubernetes Namespace in the parent AKS cluster, Key Vault, etc. A Tenant is fundamentally the following at a logical level:

**A container of resources**
All resources (except ones corresponding to infrastructure) are created within the Tenant. If we delete the Tenant, then all resources within it are terminated.

**A security boundary**
All resources within the Tenant can talk to each other. For example, a Docker container deployed in an Azure VM instance within the Tenant has access to storage accounts and SQL instances within the same Tenant. SQL instances in another Tenant cannot be reached, by default. Tenants can expose endpoints to each other, either via load balancers or explicit inter-tenant security groups and Managed Identity policies.

**User Access Control (UAC)**
Self-service is the bedrock of the DuploCloud Platform. To that end, users can be granted Tenant level access.

**Billing Unit**
Each Tenant is also a Billing Unit, so customers can see the billing dashboard, segregated by Tenants. This helps them understand the cost for each of their application deployments environments like dev, staging, and production.

**Plan**
Plans correspond to each Infrastructure. A Plan is a placeholder or template for configurations. These configurations are consistently applied to all Tenants within the plan (or Infrastructure). Examples of such configurations are:

- Certificates available to be attached to load balancers in Tenants of the Plan
- Machine images
- WAF web ACLs
- Common policies and SG rules to be applied to all resources in Tenants within the Plan
- Resource Quota. Each Plan has a resource quota that is enforced in each of the Tenants within the Plan

## Rules-based Engine

As the user submits higher-level deployment configurations via the application-centric interface, an internal rules-based engine translates the configurations to low-level infrastructure constructs automatically, while also incorporating the desired compliance standards.

## State Machine

The fundamental limitation of IaC is a serial execution of steps requiring human supervision. The DuploCloud Platform includes an intelligent state machine that applies a lower-level configuration generated by the rules engine to the cloud provider by invoking the APIs, which work asynchronously in multiple threads. Repeated failures are flagged as faults in the user interface.

## Ongoing Reconciliation

The system constantly compares the current state of the infrastructure with the desired state, which includes compliance standards and security requirements. If there is a difference, then either DuploCloud will auto-remediate or raise an alert.

# User Personas and Workflows

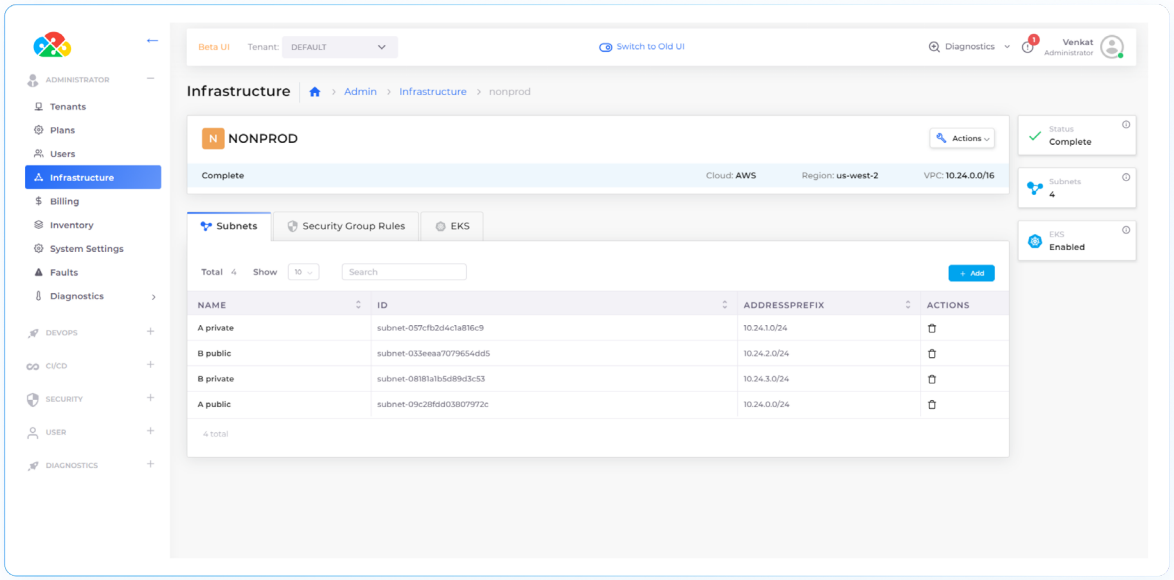
There are 4 main user personas: Administrators, Developers, Security Admins and SREs. Each persona is captured by a set of workflows and features.

# Administrators (used by DevOps)

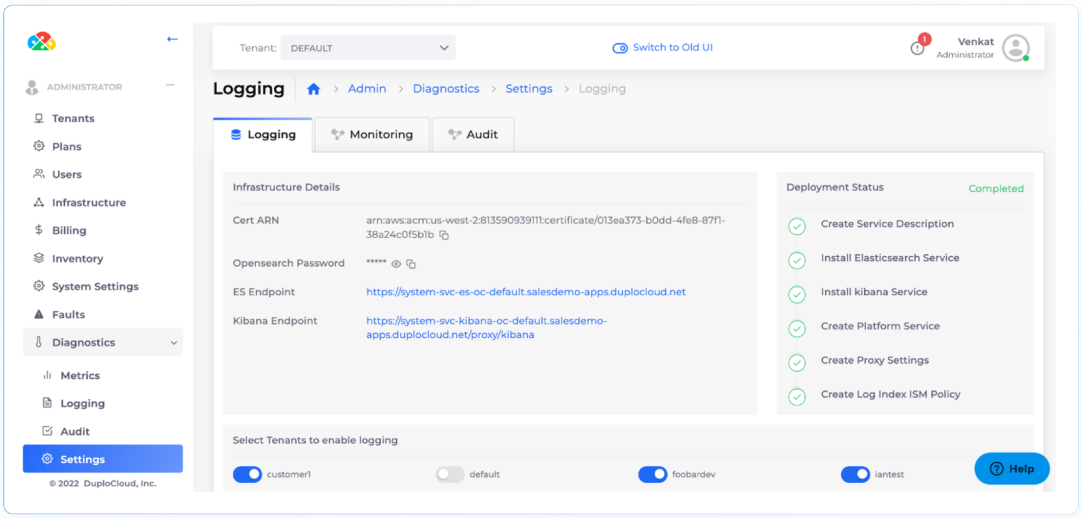
This part of the platform covers the role of the administrator, typically played by either an in-house DevOps engineer or a Team lead. There are three types of activities or workflows that involve administrators:

**Resource Provisioning**
These are resources that are relatively infrequently created and/or updated. A few examples of these are:

- Infrastructure setup that includes VPC/VNETs, subnets, Kubernetes cluster, and in case of Azure, Log Analytics, Azure Automation account, etc.
- Kubernetes upgrades



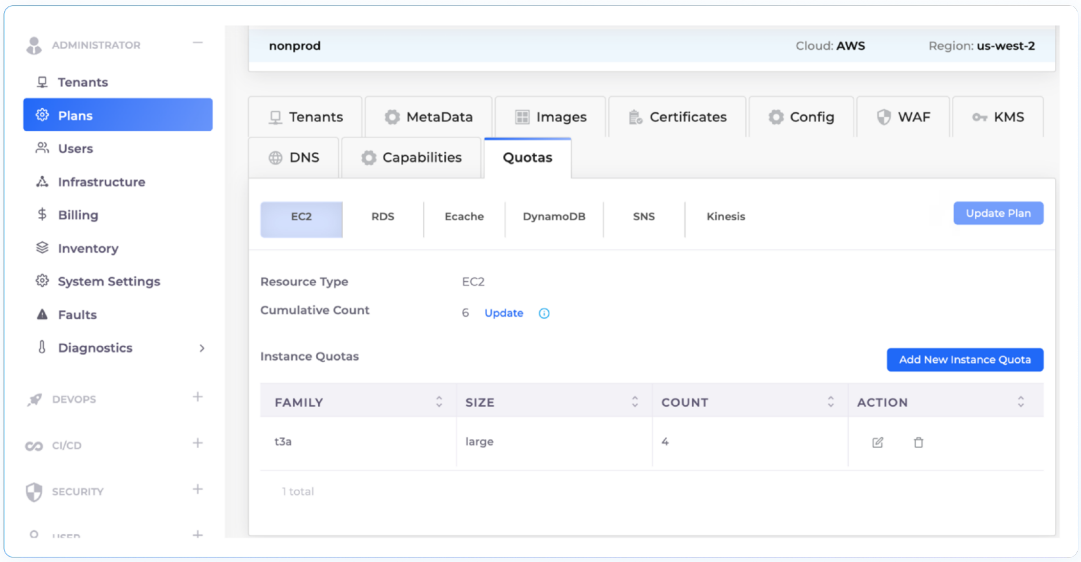
- Setup of the Centralized Diagnostics stack like Open Search, Prometheus, and Grafana used by the Tenants.



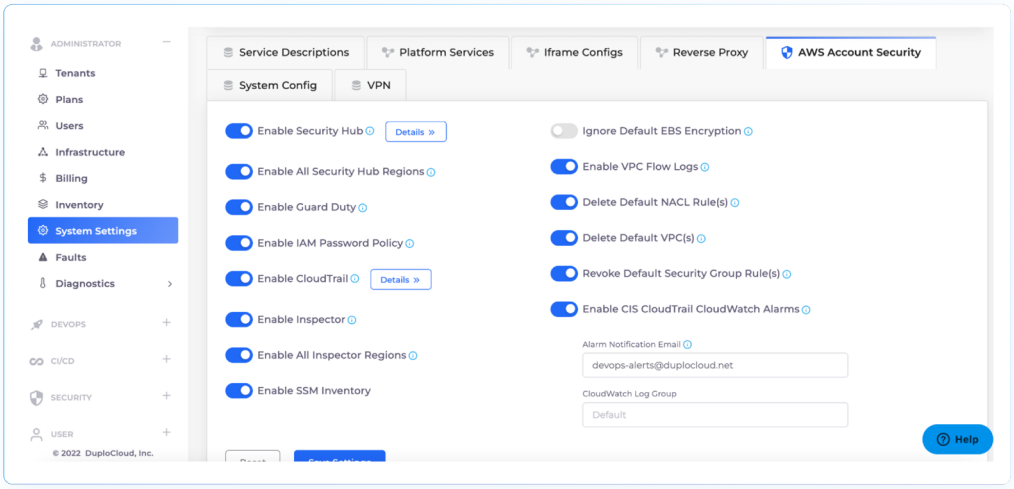

**Create resources directly in Cloud Provider and Reference them in DuploCloud**
Many resources like DNS domain, SSL Certificate, WAF Rules, and hardened Images are typically created outside of the platform. Their identifiers are added to the DuploCloud platform under the “Plan” constructs.

**User Access and RBAC**
Administrators control which users have access to what Tenants and define their roles.

**Resource Quotas**
Administrators can limit the user’s ability within the Tenant to create resources within a specific type and size.

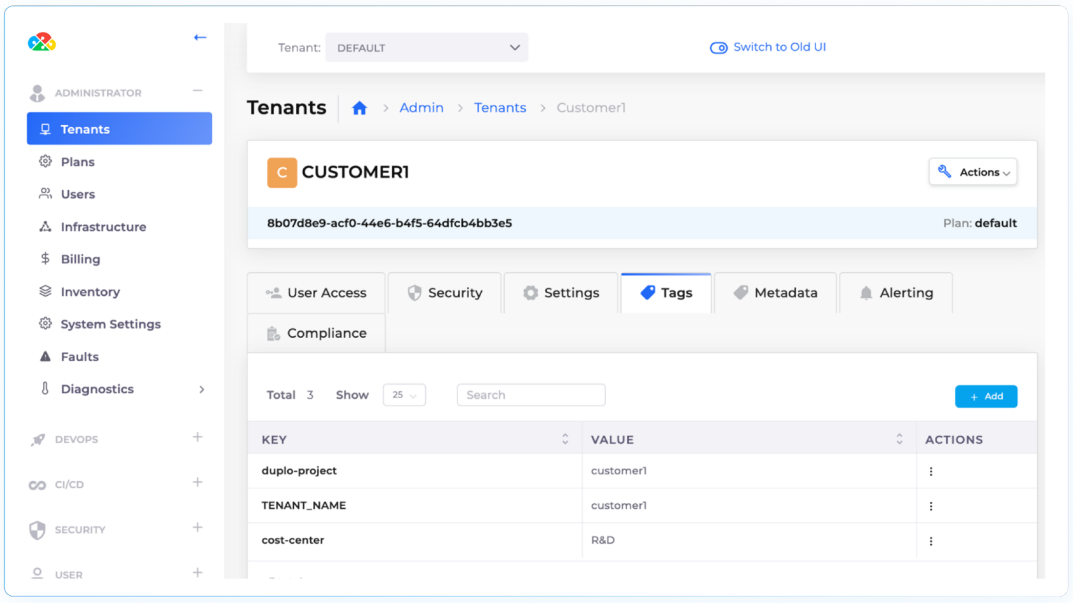


**Foundational Security Controls**
Administrators control the setup of various application-agnostic security features like AWS CloudTrail, AWS SecurityHub, Azure Defender, and others.




**Policies and Guard Rails**
There are several policies and guard rails configurable in the system. For example, blocking Tenant users from exposing public endpoints, and enforcing certain prefixes for S3 buckets and S3 bucket policies that should apply across the system.

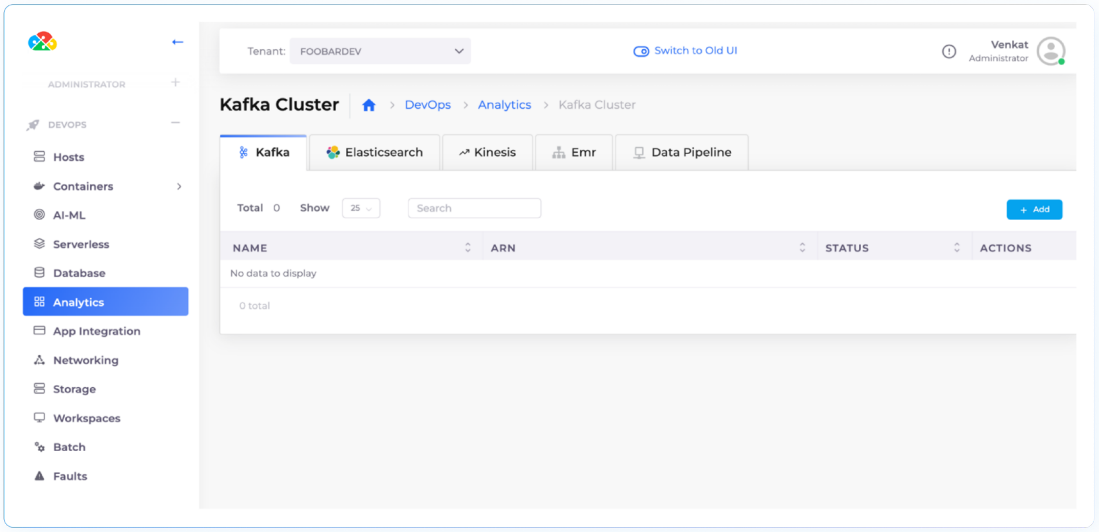
**Resource Tagging**
Administrators can set tags at the Tenant level that are automatically propagated and applied to all the resources created within the Tenant.



# Developer Role (used by Developers and Data Scientists)

Developers form the majority of our audience as DuploCloud is essentially a Developer Platform. Developers are responsible for deploying, updating, and managing their application infrastructure within a given Tenant. Each user has access to multiple Tenants and each Tenant can have multiple users. The main developer workflows are categorized as follows:

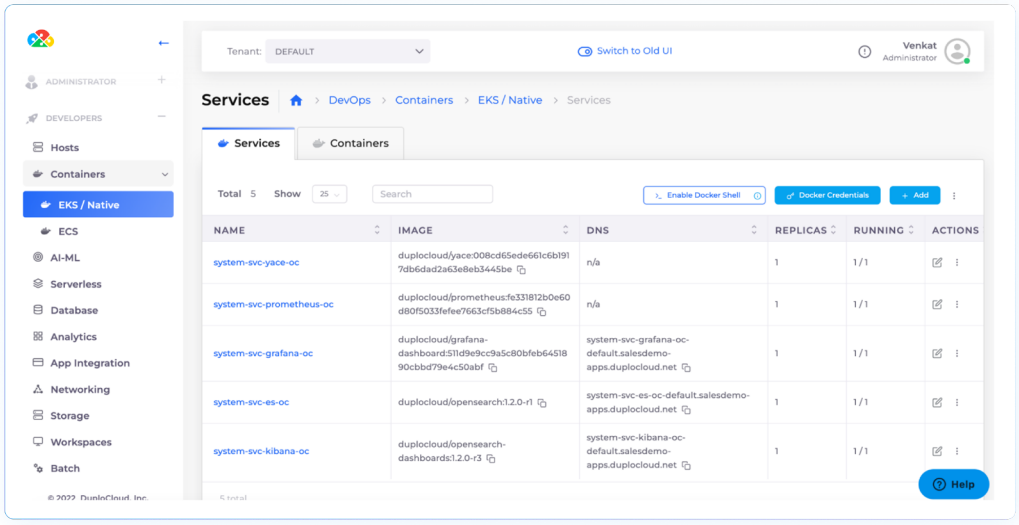
**Cloud Service Deployments**
These include dozens of cloud provider services like EC2, Azure VMs, S3, Azure blob stores, RDS, MSK, Managed Open search, SQS, SNS, Redshift, Azure DB, etc. DuploCloud supports hundreds of services. New services are added regularly. The typical turnaround time to add a cloud provider service is about a week.




**Config and Secrets Management**
Developers leverage a vast set of cloud-native services for this purpose like Kubernetes secrets and config maps, AWS SSM Param Store, Secret Store, Azure Key Vault, etc. Developers can create, update, and manage the secrets referenced by their applications without having to deal with the lower-level nuances of policies, encryption, Kubernetes drivers, etc. See the documentation page Passing Config and Secrets for more detailed information.

**Application Deployment**
Deployment patterns commonly used by Developers are:

**Docker**
DuploCloud integrates with Cloud managed Kubernetes like EKS, AKS, GKE, or cloud container orchestrators like ECS and Azure Web App. Almost all complexities of Kubernetes are hidden from the user.



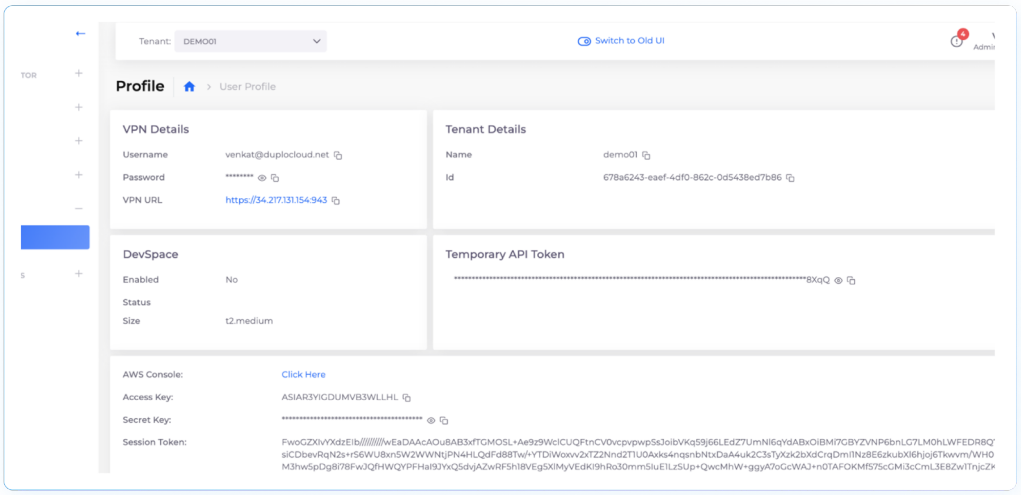

**Serverless**
Lambda, Azure Functions, and GCP cloud functions are typical serverless features that developers deploy in their applications.

**Big Data**
EMR, Apache Airflow, Glue, and Azure Databricks are examples of services data scientists use.

**AI/ML**
Sagemaker and Azure Machine Learning are examples of AI/ML services.

**Application connectivity**
Exposing applications via load balancers, ingress controllers, and API gateways that include configuring SSL certificates (provisioned by administrators).

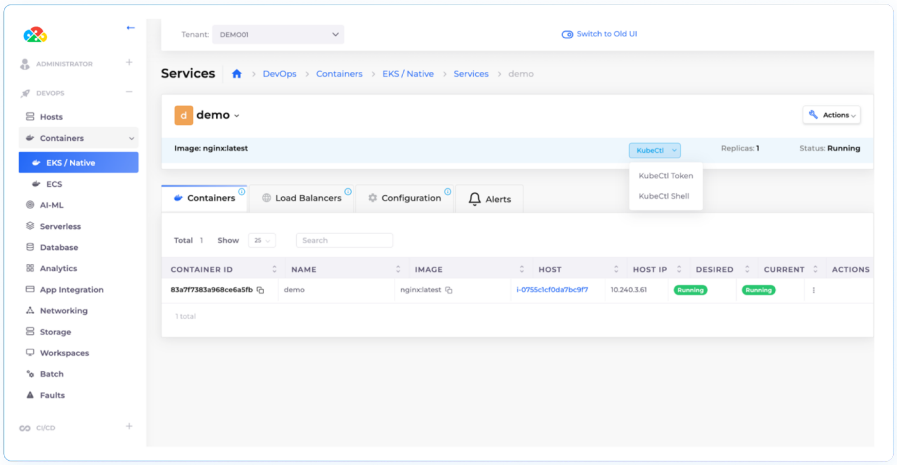
**Local Development**
Developers often need to build and test code in a local environment. They need access to cloud provider services via access keys. DuploCloud facilitates that by creating Tenant scoped keys with a limited lifetime. See the documentation page JIT Access: Access Through Command Line for more detailed information.




# Diagnostics Workflows for DevOps, Developers, and SRE Personas

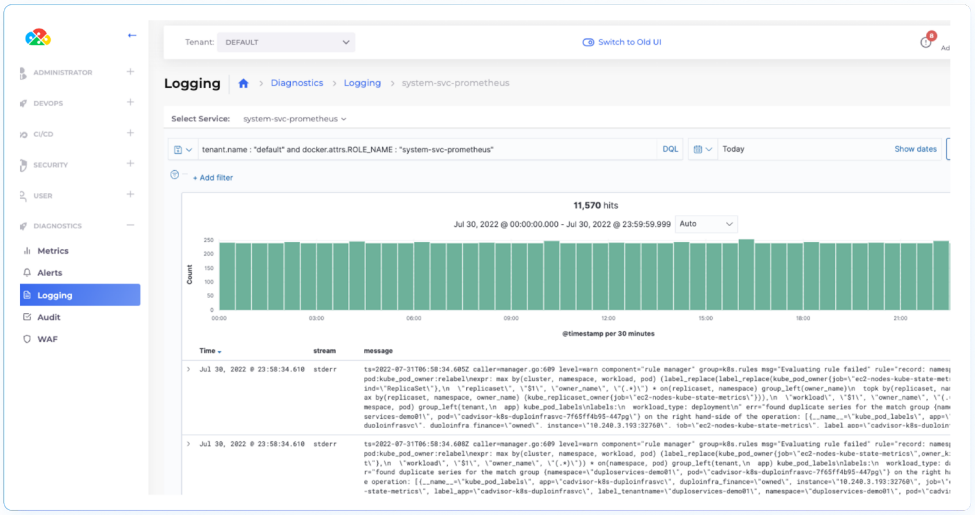
There are 4 key diagnostics functions leveraged by DuploCloud users:

**Cloud Portal, Kubectl and Shell Access**
Developers occasionally need access to direct cloud portals and services, Kubectl, and access to the application container's shell. DuploCloud creates JIT access into these systems by orchestrating underlying substrates like Kubernetes Service accounts, AWS federated login, and Azure AD. This is done on an as-needed basis using principles of least privilege; for example, when a user gets access to Kubectl, the access is scoped to the tenant's namespace only.



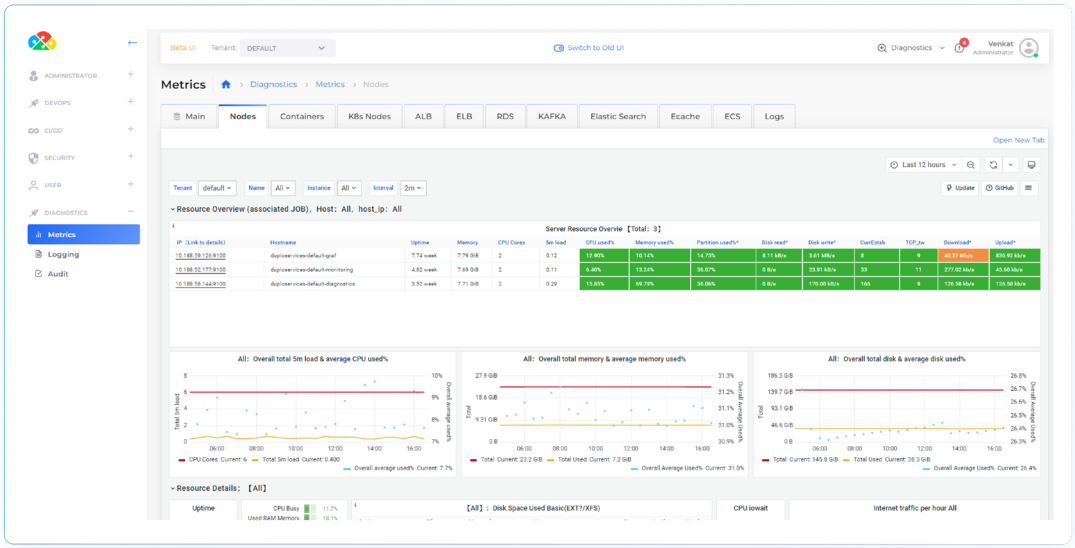
**Central Logging**

Central logging is implemented by orchestrating Elasticsearch, Kibana, and File Beat. Internally, nuances for AKS service accounts, ES ILM policy, index lifecycle and other low-level details are automated. Kibana dashboards are displayed per Tenant and per service.

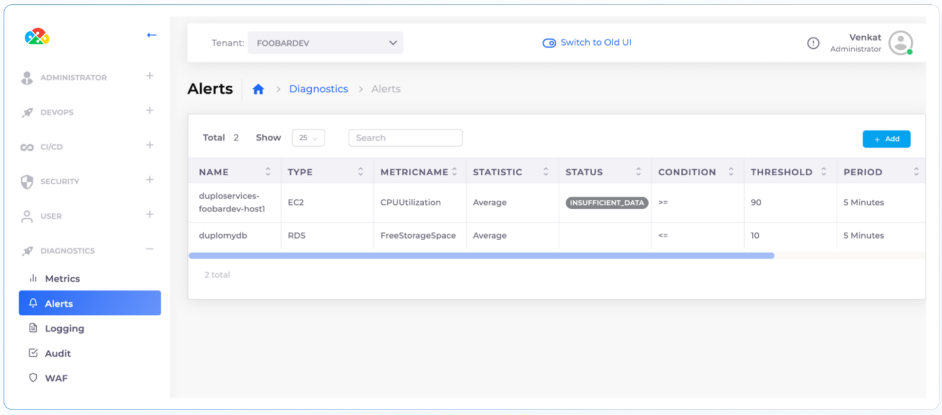



**Metrics**

Metrics are implemented using Prometheus, Grafana, and Azure monitoring with the platform managing the lower-level nuances around AKS and Azure.



**Monitoring and Alerts**
The platform is constantly monitoring the infrastructure for anomalies by default and allows the user to define custom alerts.




**Notifications**
DuploCloud consolidates all anomalies in the system, Tenant by Tenant, into the Faults sections. These notifications are sent to one of the many supported alerting tools like Sentry, PagerDuty, and New Relic.

## Security and Compliance Workflows for the SecOps Persona

Built-in best practices for various security standards are core to the DuploCloud Portal. Detailed security whitepapers describing the implementation of security controls can be found here: https://duplocloud.com/white-papers/

**Compliance Standards**
The DuploCloud platform implements compliance controls to the level of NIST 800-53, which is a superset of virtually all known standards and subsumes at the level of cloud infrastructure and most other compliance standards. More than 70% of our user base operates in regulated industries and leverages DuploCloud for the following standards:

- **SOC 2**
- **HIPAA**
- **PCI-DSS**
- **ISO**
- **GDPR**
- **NIST**
- **HITRUST**
- **Others**

**Secure by Design**
For security controls in standards like PCI and SOC 2, 70% of the controls are implemented at the provisioning of the resources and 30% of the controls are monitoring controls that are performed during post-provisioning.

*The advantage of DuploCloud being an end-to-end automation platform is that all the necessary controls are injected into the configuration automatically both at provisioning time as well as post-provisioning. This contrasts with a traditional security approach where SecOps teams get involved mostly during the post-provisioning and monitoring process.*

### Examples of Provisioning Time Controls

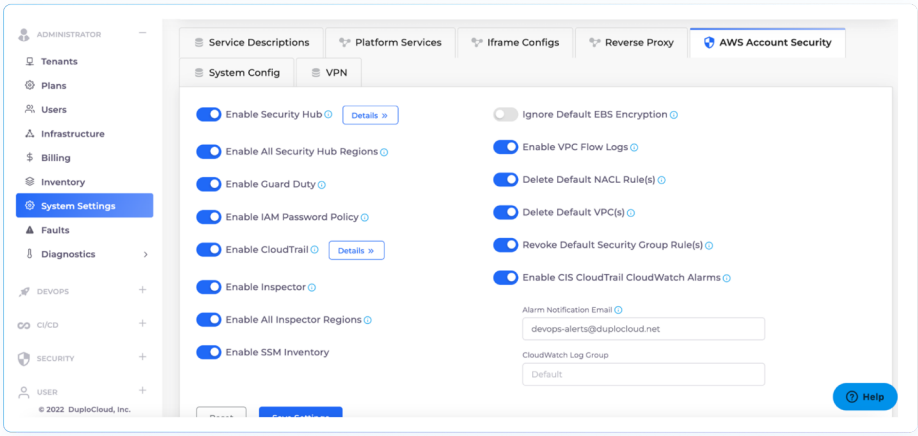
- Network Provisioning and Landing zones including VPC/VNET/VPN
- Access control roles and policies using cloud provider IAM
- Encryption-at-rest using cloud provider key management systems like KMS, Azure Key Vault, etc.
- Transport Encryption (transit), using certificates, that configures load balancers, gateways, and certificate managers
- Secrets management using secret stores like AWS secret store, Azure Key Vault, Kubernetes secrets
- Provisioning scores of cloud-native services like s3, Dynamo, Azure storage, Kafka, OpenSearch, etc. Provisioning includes configuring and connecting various access policies, availability considerations, scale, and of course various compliance configurations. For example, during S3 setup, the system manages SSE, public access block, versioning (when needed), and IAM access control among other things.

### Examples of Post Provisioning Time Controls

- Vulnerability Detection
- CIS benchmarks
- Cloud Vulnerability and Cloud trail Monitoring.
- File Integrity Monitoring
- Host and Network Intrusion Detection
- Virus Scanning and Malware detection
- Inventory management
- Host Anomaly Detection
- Email Alerting
- Incident Management

For a detailed list of security controls, categorized by standards, check out our white papers at https://duplocloud.com/white-papers/

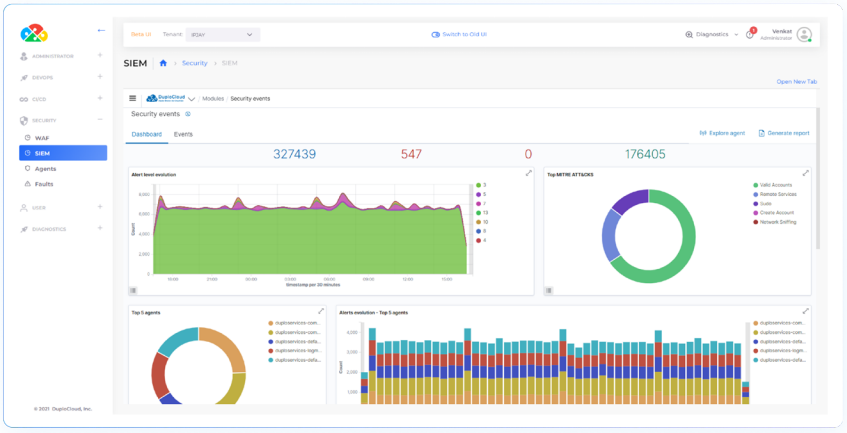
**Foundational Guard Rails and System Setup**
Security features like AWS CloudTrail, AWS SecurityHub, Azure Defender, AWS GuardDuty, as well as baseline policies, can be turned on with a click, as shown below.




**SIEM (Security Incident and Event Management)**
SIEM is a centralized system that aggregates and processes all events. DuploCloud uses open-source Wazuh as a SIEM and this is orchestrated and integrated into the workflows. The primary functions of the system are:

- Data Repository
- Event Processing Rules
- Dashboard
- Events and Alerting

Distributed agents of this platform (Ossec Agents) are deployed at various endpoints (VMs in Cloud), where they collect event data from various logs like syslogs, virus scan results, NIDS alerts, file Integrity events, etc. Data is sent to a centralized server and is processed using a set of rules to produce events and alerts that are stored in Elasticsearch where dashboards can then be generated. Data can also be ingested from sources like AWS CloudTrail, AWS Trusted Advisor, Azure Security Center, and other non VM-based sources.

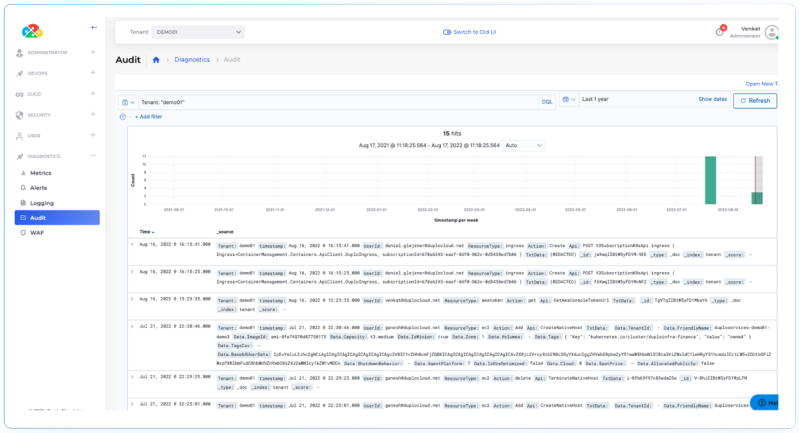


**Agent Modules**

For many of the security features, several agent-based software packages are installed in each in-scope VM. A few examples are the Wazuh agent, used to fetch all the logs; the ClamAV virus scanner, the AWS Inspector, which provides vulnerability scanning; and Azure OMS and CloudWatch agents for host metrics. While these agents are installed by default, DuploCloud provides a framework in which the user can specify an arbitrary list of agents in the respective format and the DuploCloud software will install these automatically in any launched VM. If any of these agents crash, DuploCloud sends an alert. You can also integrate with your own XDR, SIEM, and other solutions by leveraging this feature for agent installation.

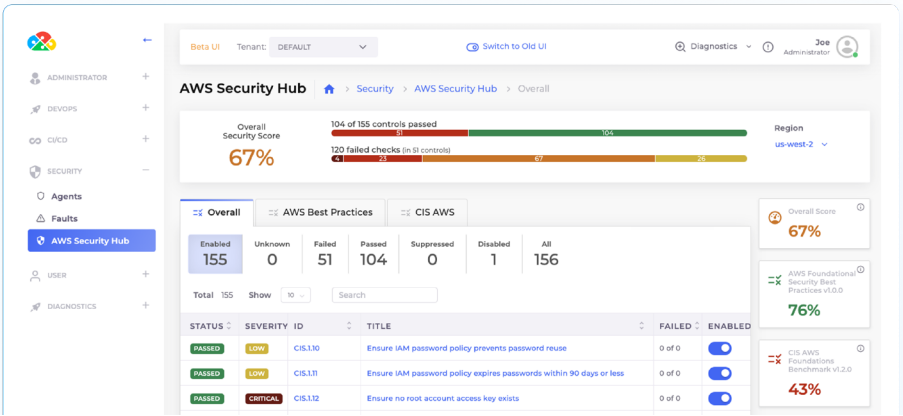
**Audit Trails in Application Context**

When using raw IaC without a management system like DuploCloud, DevOps teams build cloud deployment from an operations and infrastructure perspective, rather than from the application perspective. Many times resources are not appropriately tagged with an application context and if you require an audit trail at the cloud provider level, as with AWS CloudTrail or Azure event logs, it can be hard to correlate to the application. In DuploCloud, audit trails are available per Tenant with detailed metadata in the trails in an application-specific context.




**AWS SecurityHub and Azure Defender**

DuploCloud integrates natively with cloud provider-native solutions like AWS Security Hub and Azure Defender that includes setup, management, and operations.



## Inventory

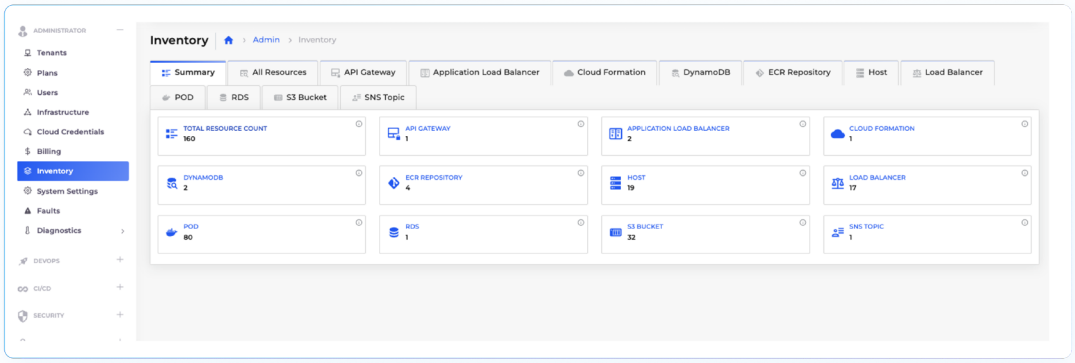
Inventory management is a key element of security and cost management, as well as a compliance need. The DuploCloud platform manages inventory at three levels:

### Tagging

By default, all resources are tagged by Tenant name and the custom tags set by the user at the Tenant level. When new resources are created within the Tenant, all tags are automatically propagated to all the underlying resources associated with the Tenant.

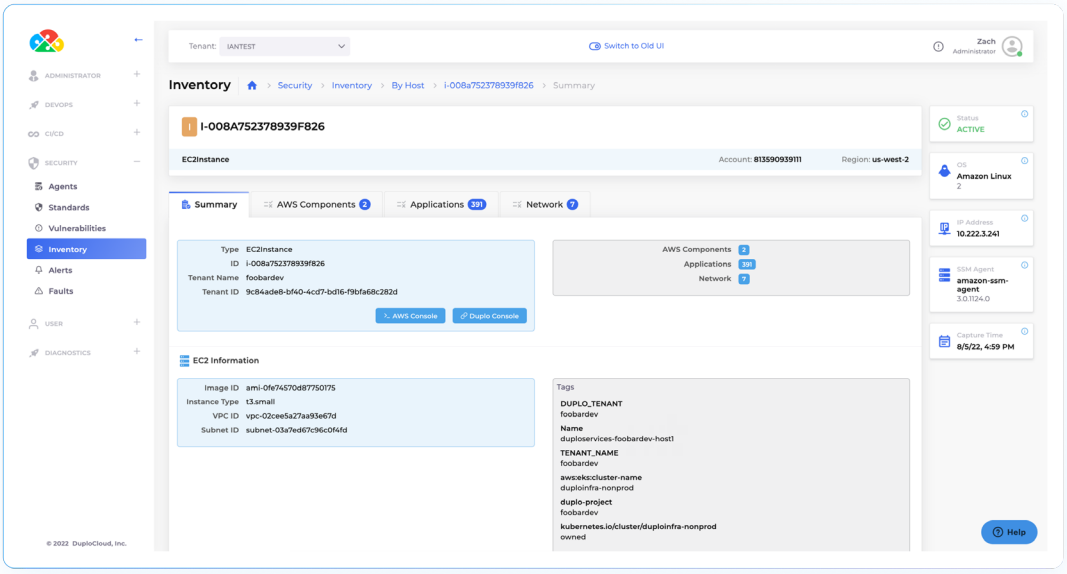
### Cloud Inventory

DuploCloud provides a catalog of all the resources in an application-centric view as well as a flat cloud service view.



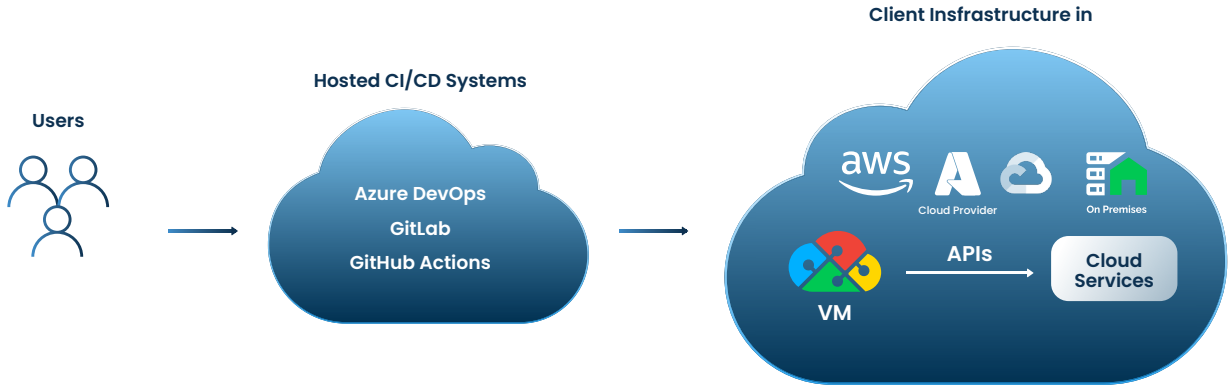

### VM Inventory

OS-level inventory is pulled through the SIEM, as well as cloud provider solutions like AWS Inspector or Azure Mon agent.



# Continuous Integration and Deployment (CI/CD)

CI/CD is a layer on top of DuploCloud and any CI/CD system like Jenkins, GitHub, GitLab, and Azure DevOps can seamlessly integrate with DuploCloud by either calling our REST APIs or via Terraform. You build your pipelines and CI/CD workflows in these CI/CD systems that invoke DuploCloud software via APIs or Terraform, as shown in the figure below.




DuploCloud creates prepackaged libraries and modules to invoke DuploCloud functionality from CI/CD systems like GitHub actions.

Refer to our documentation at https://docs.duplocloud.com/docs/ci-cd/github-actions

Following are the typical integration points between CI/CD systems and DuploCloud:

**Cloud Access for Hosted Runners**
Builds are executed in the CI/CD platform's SaaS infrastructure and outside of the organization's infrastructure. For the builds to reach the infrastructure they need either credentials or VPN access. DuploCloud's Platform facilitates this by providing JIT (Just-in-Time) access scoped to Tenants for the build pipelines. Users create a "CICD" user in the DuploCloud portal that has limited access to the desired Tenants. A token is created for the user and added to the CI/CD pipelines. The most common example of a workflow is when one builds a Docker image and pushes the Docker image to the Cloud Provider registry. Access to the cloud provider registry is facilitated via DuploCloud.

**Deploying Self-Hosted Runners within the Tenant**
A set of build containers are deployed within the same Tenant as the application itself. This allows the build to seamlessly access the Tenant's resources as if it were the application and includes Docker registries, internal APIs, object stores, SQL, etc.

**Deployment of new Builds**
Within the deployment step, once the Docker image has been built, the build script invokes DuploCloud's service update API with Tenant ID, Service Name, and Image ID as parameters. DuploCloud Platform executes the deployment, using the same API that the DuploCloud UI calls when a user updates a service image via the DuploCloud API.

**Status Checks**
In the CI/CD pipelines after a certain build has been deployed, the pipeline invokes the DuploCloud API to get the overall status of the services.

**Environment Create, Delete, and Update**
Some use cases involve bringing up a whole new environment by triggering a certain pipeline that executes a Terraform script invoking the DuploCloud Platform to deploy the whole environment. Similarly, it can be destroyed by a user trigger of the pipeline.

# Conclusion

DuploCloud delivers an Integrated Developer and DevOps Platform out-of-the-box, so organizations don't have to build it themselves by writing thousands of lines of code over many months and years.

Developers can build, deploy, and manage applications in a self-service manner, within the guard rails defined by the Platform Engineering and Security teams. Compliance controls and best security practices are built in. DuploCloud's greatest advantage is in enabling self-responsibility for engineers, without requiring them to be subject matter experts in operations infrastructure and security. Our platform allows developers to take services and apps from idea to production on their own. This drives accountability, as product teams are now responsible for the configuration, deployment, or rollback process. Increased visibility, and monitoring allow teams to collaborate better and troubleshoot faster.

Duplocloud's DevOps Automation Platform is the world's first IDP that supports multiple clouds and handles security and compliance, enabling Platform Engineering teams to provide self-service to developers.

Learn more about how DuploCloud reduces migration costs by 75% and speeds up your organization's deployment times by a factor of ten.

**Contact us today for a personalized one-on-one walkthrough, and see DuploCloud in action for yourself.**