Technology Startup

Cloud and Development Operations Blueprint

Kris Musard - July 2023

Abstract

The following document outlines a blueprint for efficiently bootstrapping SOC2 compliant cloud and development operations for a technology startup. The technology stack presented makes use of the following:

- ArgoCD (Deployment)
- AWS (Cloud Provider)
- Azure Active Directory (Identity Provider)
- DataDog (Monitoring & Observability)
- GitHub Actions (Code Repository & CICD)
- Jira (Agile Workflow & Release Reporting)
- Kubernetes/EKS (Infrastructure/Deployment)
- Terraform (Infrastructure as Code)

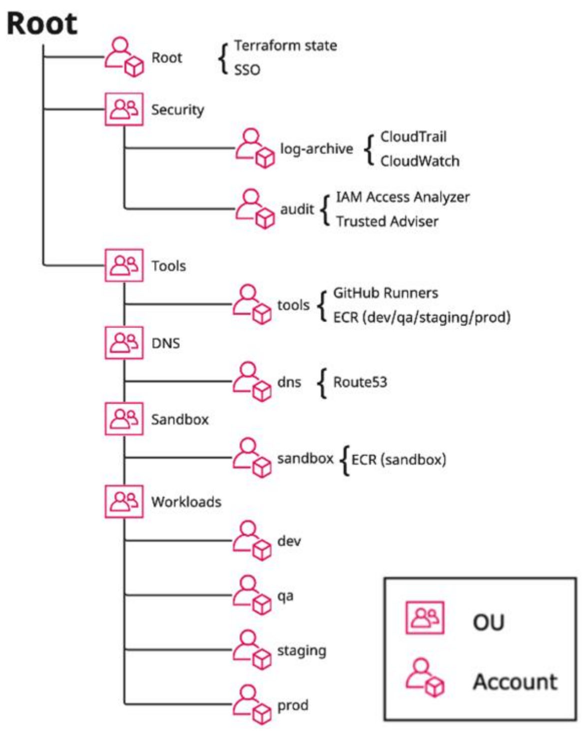
The design relies on agnostic guidance for management of infrastructure, development, and cyber security such as “The Twelve-Factor App” (https://12factor.net/), “Trunk Based Development” (https://trunkbaseddevelopment.com/), and “Minimum Viable CD” (https://minimumcd.org/minimumcd/). This allows flexibility to port the design to different combinations of cloud providers, CICD platforms, container orchestration tools, or alternate deployment models such as serverless.

Day 0 – Planning & Strategy

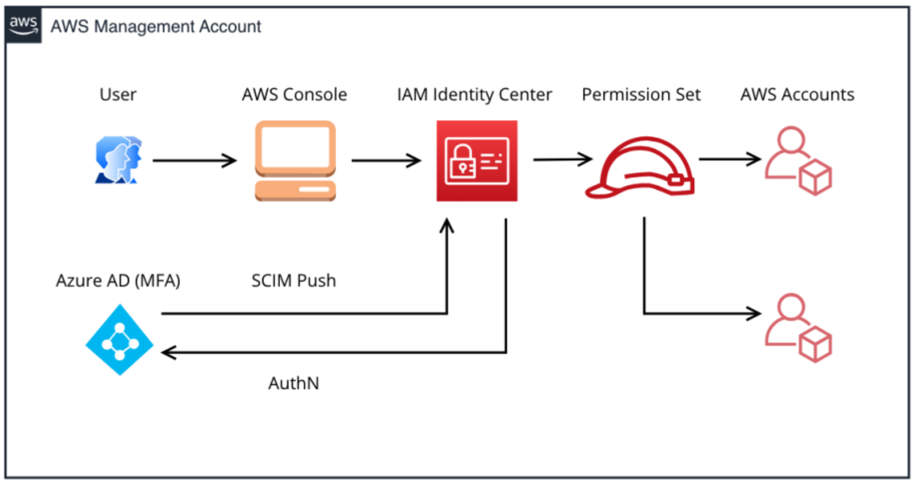
1. Resources – The execution of this strategy will require a minimum of one Lead DevOps Engineer / SRE. It will take approximately 800 resource hours to complete. Additional resources may be added to shorten the calendar window for go-live down to 1-2 months.
2. Buy Commodity, Don’t Build It – Focus on our core product while leveraging SAAS providers to host CICD, Monitoring, and Agile Workflow tools. This can be adjusted for cyber security, regulatory, or other concerns that require a particular system to be self-hosted.
3. 100% Automation – Enforce automation of all operations from the beginning.

Day 1 – Infrastructure Rollout

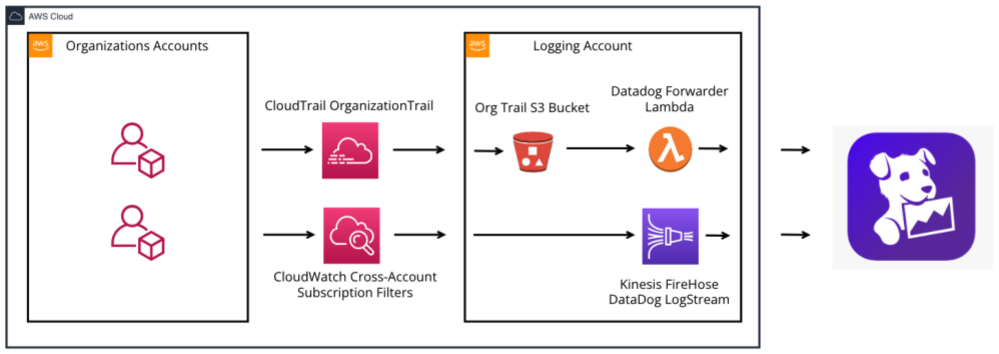
1. Activate accounts with cloud providers and SAAS vendors.
2. Setup Jira Kanban/Scrum, populate epics for phase 0 rollout, schedule daily standups and agile planning meetings. This will allow us to measure progress and track project completion.
3. Create/populate application code repositories.
    a. Development staff may begin local development of application code and tests.
4. Create resource accounts on the cloud provider.



5. Configure SSO for administrators and engineering staff.

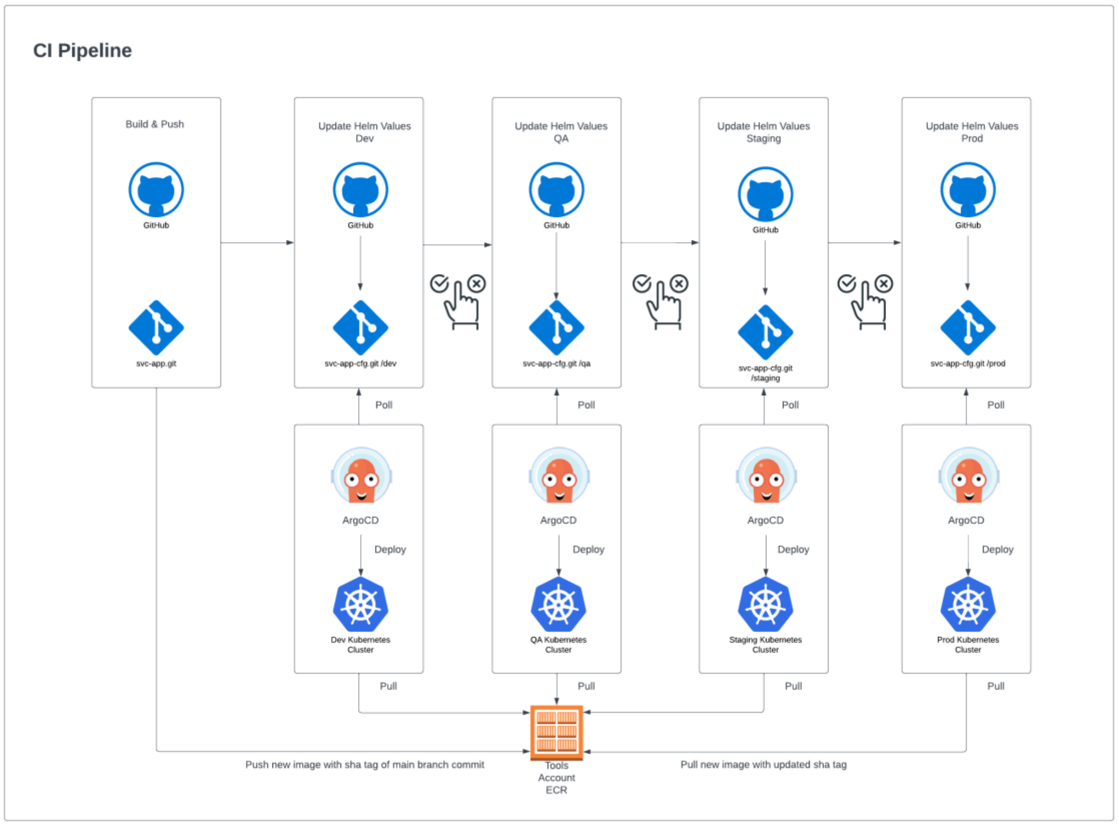


6. Bootstrap Infrastructure as Code (IaC).
    a. Configure centralized terraform state.
7. Bootstrap monitoring environments.
    a. Configure connectors from AWS, GitHub, and other platforms in the environment.
8. Deploy cloud infrastructure using IaC.
    a. Includes agents and configurations for centralized monitoring across all components.



AWS Cloud
Organizations Accounts
CloudTrail OrganizationTrail
CloudWatch Cross-Account Subscription Filters
Logging Account
Org Trail S3 Bucket
Datadog Forwarder Lambda
Kinesis FireHose DataDog LogStream

    b. Includes cloud security configurations and authorizations for the external CICD system.
9. Configure CICD and application deployment



10. Train development and operations staff. This can be conducted in parallel ahead of go live and will continue as new tools and functionality are introduced.

Day 2 – Ongoing Operations and Improvements

Development Operations

At go live, the development process will make use of the following GitHub workflows triggered from the application code repositories:

1. pr.yaml
    a. Triggered by pull requests to main.
    b. Runs tests for code functionality, quality, security, and compliance before merging to the main branch.
    c. Requires 2 approvals.
    d. FUTURE: Enhance pre-merge testing by configuring automated PR review environments using the Argo PR builder.
2. ci.yaml
    a. Triggered by pull requests merging to main.
    b. Builds image and pushes to a central ECR repository shared to dev/qa/staging/prod accounts.
    c. Updates the dev folder in a corresponding configuration repo. This folder stores helm and kustomize configurations polled by ArgoCD running on each Kubernetes cluster. ArgoCD polls the config repo change within 60 seconds and executes the update on the cluster.
    d. The pipeline then pauses with manual approval gates between QA/Staging/Prod, which repeat the config repo update process for each environment folder.
    e. FUTURE: Over time, replace manual gates and testing with automated testing allowing the pipeline to confidently execute through production in an automated fashion.
3. update-sandbox.yaml
    a. Manually triggered by engineering staff to test and deploy code in progress to a live environment.
    b. Runs against a sandbox cluster which uses a separate dedicated sandbox ECR repository.
    c. FUTURE: The automated PR review environments mentioned above will provide this functionality, allowing developers to spin up preview environments for personal needs under a draft PR.
4. hotfix.yaml
    a. Variation of the normal ci workflow which deploys a hotfix branch as opposed to the main branch.
    b. Bypasses dev & qa, and goes straight to staging before manual approval to prod.

Infrastructure Operations

To be continued…