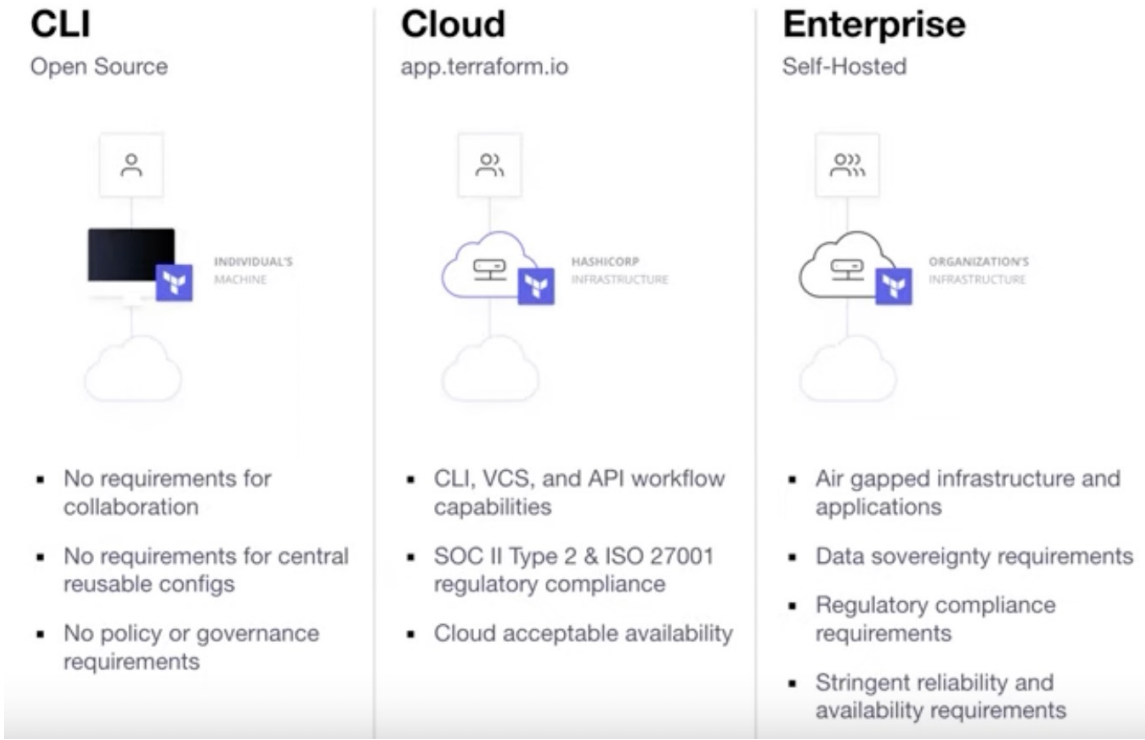
Terraform Automation
Terraform Cloud

Terraform Cloud

Terraform Cloud

- As an alternative to home-grown automation solutions, Hashicorp offers Terraform Cloud.
- Internally, Terraform Cloud runs the same Terraform CLI commands described above, using the same release binaries offered for download on this site.
- Terraform Cloud builds on the core Terraform CLI functionality to add additional features such as role-based access control, orchestration of the plan and apply lifecycle, a user interface for reviewing and approving plans, and much more.
- It will always be possible to run Terraform via in-house automation, to allow for usage in situations where Terraform Cloud is not appropriate. It is recommended to consider Terraform Cloud as an alternative to in-house solutions, since it provides an out-of-the-box solution that already incorporates the best practices described in this guide and can thus reduce time spent developing and maintaining an in-house alternative.



Terraform Cloud

- Main documentation link
- Hosted SaaS at https://app.terraform.io.
- Main goals / benefits
  - Manages Terraform runs in a consistent and reliable environment
  - Easy access to shared state and secret data
  - Facilitate access controls for approving changes to infrastructure
  - Private registry for sharing Terraform modules
  - Detailed policy controls for governing the contents of Terraform configurations
- Tiers
  - Free (small teams) : connect Terraform to version control, share variables, run Terraform in a stable remote environment, and securely store remote state.
  - Paid tiers allow you to add more than five users, create teams with different levels of permissions, enforce policies before creating infrastructure, and collaborate more effectively.
- Terraform Enterprise: self hosted version of Terraform Cloud

Terraform Cloud (cont.)

- Foundations of Terraform Cloud Workflow
  - Remote Terraform execution (local also available)
  - Workspace-based organizational model
  - Version control integration – integrate with GitHub, Bitbucket, Gitlab, Azure Devops
  - Command-line integration (optional)
  - Remote state management with cross-workspace data sharing
  - Private Terraform module registry.

Terraform Cloud Workspaces

- Terraform Cloud manages infrastructure collections with *workspaces* instead of directories.
- A workspace contains everything Terraform needs to manage a given collection of infrastructure, and separate workspaces function like completely separate working directories.
  - **IMPORTANT**: Terraform Cloud Workspaces are different from Terraform CLI Workspaces (allow sharing multiple states under a single directory)
- Additional Content in workspaces
  - **State versions:** Each workspace retains backups of its previous state files. Although only the current state is necessary for managing resources, the state history can be useful for tracking changes over time or recovering from problems. (Terraform State in Terraform Cloud)
  - **Run history:** When Terraform Cloud manages a workspace's Terraform runs, it retains a record of all run activity, including summaries, logs, a reference to the changes that caused the run, and user comments. (Viewing and Managing Runs)

Component	Local Terraform	Terraform Cloud
Terraform configuration	On disk	In linked version control repository, or periodically uploaded via API/CLI
Variable values	As `.tfvars` files, as CLI arguments, or in shell environment	In workspace
State	On disk or in remote backend	In workspace
Credentials and secrets	In shell environment or entered at prompts	In workspace, stored as sensitive variables

Terraform Cloud and Version Control Systems (VCS)

Terraform Cloud and VCS Providers - Intro

- General docs on integrating TF Cloud with Version Control Systems
- GitHub
  - GitHub.com – "configuration-free" : uses a preconfigured GitHub App (only with GitHub.com)
  - GitHub.com (OAuth)
  - GitHub Enterprise
- Azure DevOps Server
- Azure DevOps Services (Oauth)
- GitLab.com
- GitLab EE and CE
- Bitbucket Cloud
- Bitbucket Server

General Concepts for OAuth-based Integration

On your VCS	On Terraform Cloud
Register your Terraform Cloud organization as a new app. Get ID and key.	
	Tell Terraform Cloud how to reach VCS, and provide ID and key. Get callback URL.
Provide callback URL.	
	Request VCS access.
Approve access request.	

- Uses OAuth protocol to Authenticate with VCS
- Can use multiple VCS connections – one selected when creating a Workspace
- Terraform Cloud uses webhooks to monitor new commits and pull requests in the VCS
  - When someone adds new commits to a branch, any Terraform Cloud **workspaces** based on that branch will begin a Terraform run. Usually a user must inspect the plan output and approve an apply, but you can also enable automatic applies on a per-workspace basis. You can prevent automatic runs by locking a workspace.
  - When someone submits a pull request/merge request to a branch, any Terraform Cloud workspaces based on that branch will perform a speculative plan with the contents of the request and links to the results on the PR's page. This helps you avoid merging PRs that cause plan failures.
- VCS Event viewer – Beta - currently only for GitLab.com connections

Terraform Cloud and GitHub

Terraform Cloud VCS - GitHub

- Two Mechanisms
  - Configuration Free (TF recommends it for beginners)
  - OAuth 2.0 (required for more advanced features like registries)

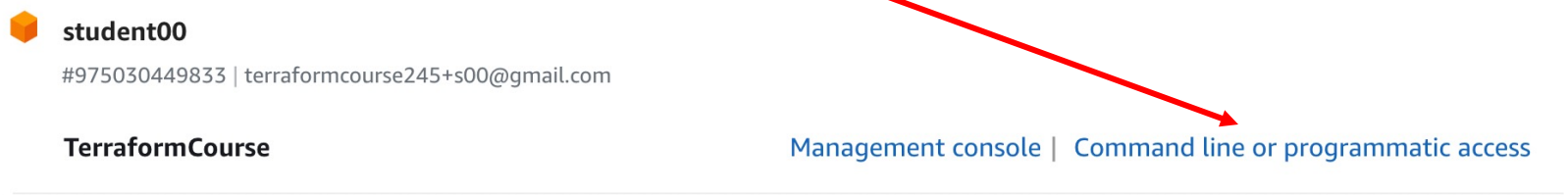
Lab: Terraform Cloud and GitHub (1)

```
echo "# tf-cloud-01" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:rpgd60/tf-cloud-01.git
git push -u origin main
```

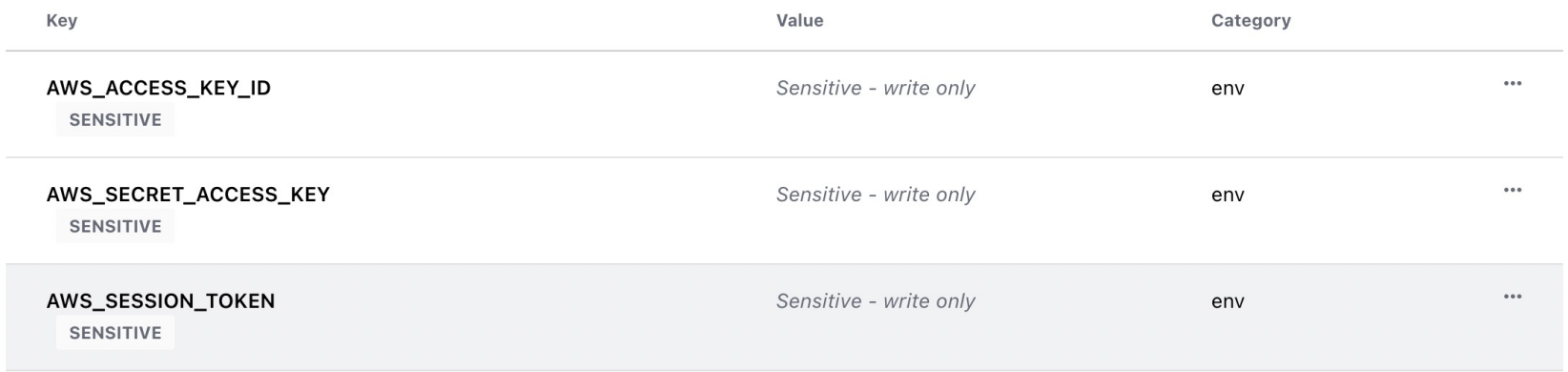
- In GitHub, create a repo: tf-cloud-01
  - Repo as created in GitHub should be empty (no README, no .gitignore)
  - Perform actions to initialize a repo in your PC (outside of course “official” repo) – Just as we did yesterday for the s3 module.
- Create a TF Cloud Account (https://app.terraform.io)
  - Create an organization (e.g. tf-course-2303)
  - Create a Project (drop down menu on right): “Project1”
  - Create a workspace
    - Type : Version Control (1st option)
    - Connect to VCS -> GitHub (you may need to authenticate with GitHub)
    - Select the repo you just created (tf-cloud-01)
    - Select the Project1 you created earlier
    - Click Create – this will fail if the repository is empty or in the process of being created

Lab: Terraform Cloud and GitHub (2)

- Configure environment variables for Access to AWS – This is for TF Cloud to be able to authenticate with AWS
  - Go to your AWS SSO page, login, and get the Access Key and Access Secret Key. Copy the values to a text file – DO NOT put in any Repo



- Back in Terraform cloud, create three ENVIRONMENT variables with the values you just got from AWS SSO (note “env”) in image below



Lab: Terraform Cloud and GitHub (3)

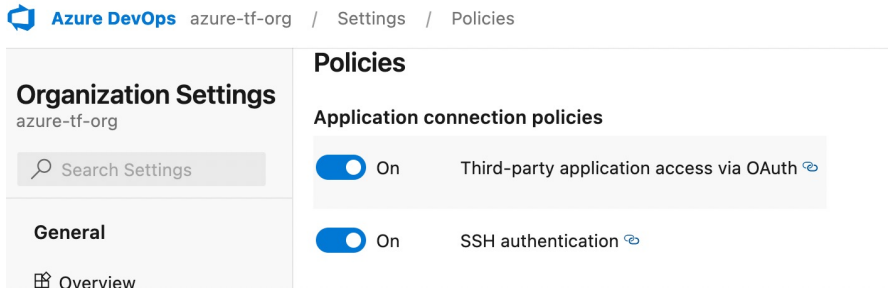
- Add Terraform code to the repo
  - You may now need to work with two VS code windows
    - One to get code from “course repo” - the lab descriptions
    - One for the tf-cloud-01 repo
  - Put some TF code in your tf-cloud-01 repo
    - Copy all .tf files from lab_12a_basic
    - Paste them to your tf-cloud-01 repo
  - Commit and push code to GitHub (git add, git commit, git push
- Go back to TF Cloud and ”trigger run” (actions)
  - Select “plan only”
  - Select TF version “1.4.1”
- Review Plan
- Apply

Lab: Terraform Cloud and GitHub (4)

- Make some modification in code
  - For example add a tag to ec2 instance
  - Commit and push change
  - TF Cloud should pick it up and offer a plan
- Explore state in Terraform Cloud

Terraform Cloud and Azure DevOps Repos

Terraform Cloud with Azure DevOps Repos



- Integration Docs
- Uses OAuth
- Summary of Steps
    0. In Azure DevOps 3rd party access (OAuth) at organization level
    1. Create a new connection in Terraform Cloud and get the callback URL.
    2. On your VCS, register your Terraform Cloud organization as a new application. Provide the callback URL and get the application ID and key.
    3. Provide Terraform Cloud with the application ID and key. Then, request VCS access.
    4. On your VCS, approve the access request from Terraform Cloud.

Terraform Cloud – Private Registry

Terraform Cloud - Private Registry

- Documentation
- Public Providers and Modules
  - Synchronize Public Providers/Modules between public registry and organization's public registry
  - Basically "centralizes" in your private registry access to the public providers/modules you are interested on
- Private Providers and Modules
  - Developed by your organization

- Lab / Exercise : publish your own module using GitHub or Azure Repos

Automating deployment of Terraform Cloud

Automating Deployment of Terraform Cloud

- Terraform Cloud has its own API - It is thus possible to use Terraform itself to “automate” Terraform Cloud
- Terraform Enterprise Provider: tfe (source in github)
- Examples
  - Deploy organization and workspace(s)
  - Deploy variable set

```
terraform {
  required_providers {
    tfe = {
      source  = "hashicorp/tfe"
      version = "~> 0.36.0"
    }
  }
}

provider "tfe" {
  hostname = "app.terraform.io"
  token    = var.token
}
```

```
resource "tfe_organization" "test" {
  name  = "org-tfe-1"
  email = "alice@example.com"
}

resource "tfe_variable_set" "test" {
  name         = "common-acme"
  description  = "Common Variables for ACME"
  global       = false
  organization = tfe_organization.test.name
}

resource "tfe_variable" "region" {
  key             = "region"
  value           = var.region
  category        = "terraform"
  description     = "Location for RGs"
  variable_set_id = tfe_variable_set.test.id
}
```

Terraform Cloud tfe provider - resources

- Video - Provisioning Resources using Terraform Cloud with an API-Driven Workflow
- Video : “Terraforming” Terraform Cloud

Alternatives to Terraform Cloud

Alternatives To Terraform Cloud

- Good comparison analysis (July 2022)
- Terraform Cloud vs:
  - Spacelift
  - Env0
  - Scalr
  - Cloudify
  - Atlantis
- Criteria

- See also TerraKube (recent 2022 tool, not included in the analysis above)

	Terraform Cloud	Atlantis	Spacelift	Env0	Scalr	Cloudify
On-Premise Support	✅	✅	❌	❌	✅	⭐
Non-TF IaC Support	❌	❌	✅	✅	❌	⭐
Policy Enforcement	✅	✅	✅	✅	⭐	✅
Chained Workspaces	❌	❌	⭐	✅	✅	❌
Terraform Module Registry	✅	❌	⭐	✅	✅	✅
Self-Service Infrastructure	✅	❌	❌	✅	✅	✅
Customizable Build Steps	⚠️	✅	✅	⭐	✅	⭐
Cost Calculation	✅	❌	✅	⭐	✅	✅
Infrastructure Visibility	❌	❌	⭐	❌	❌	✅

Alternatives To Terraform Cloud (2)

- Good comparison analysis (July 2022)
- Terraform Cloud vs:
  - Spacelift
  - Env0
  - Scalr
  - Cloudify
  - Atlantis
- Criteria

- See also TerraKube (recent 2022 tool, not included in the analysis above)

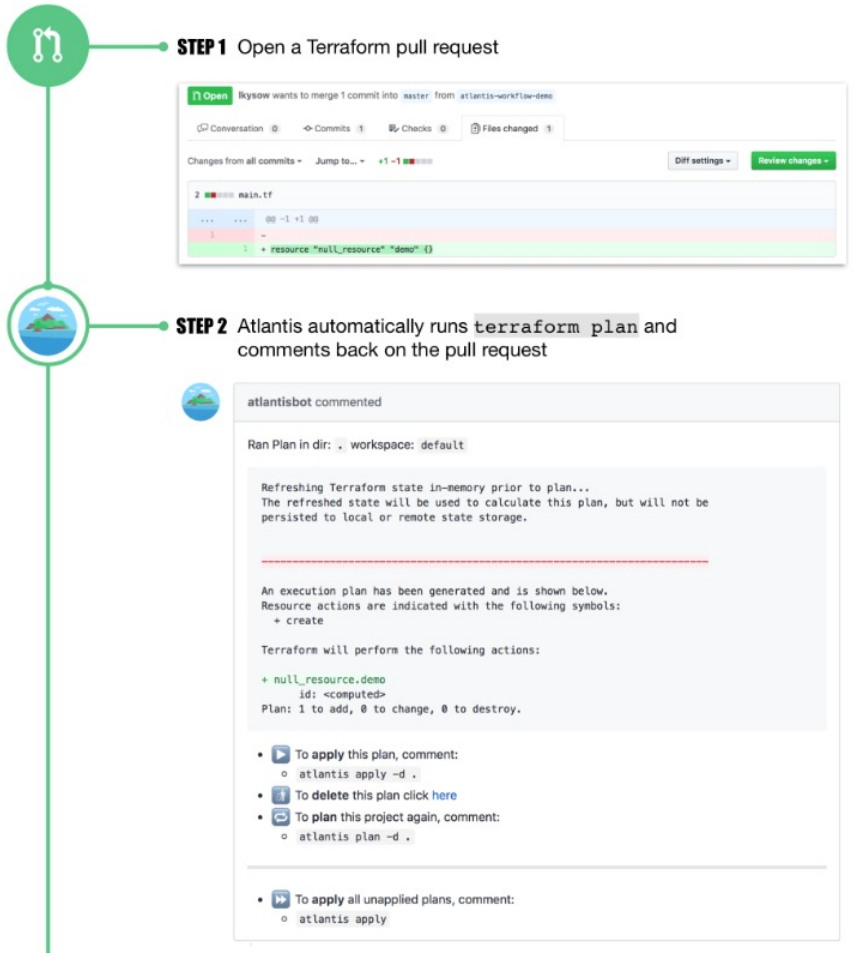
	Justification for stars
On-Premise Support	**Cloudify** has the best architecture with the flexibility to start simple and then distribute.
Non-TF IaC Support	**Cloudify** supports the most integrations by far, enabling you to capture your entire environment instead of just part of it.
Policy Enforcement	**Scalr** has the best in-app support for policy modifications that affect hundreds of workspaces.
Chained Workspaces	**Spacelift** has the best UI for visualizing chained workspaces, including great options for filtering.
Terraform Module Registry	**Spacelift** is the only product that built first-order support for automated module testing as part of a chained workspace rollout.
Self-Service Infrastructure	Each of the four has relatively comparable offerings within its own niche. There is no clear winner.
Customizable Build Steps	Both **Env0** and **Cloudify** have extensive abilities to customize. Cloudify is the most complex but the most flexible. Env0's is more straightforward but less flexible.
Cost Calculation	**Env0** integrates directly with cloud providers to give the actual running cost of each workspace. It also integrates with infracost for projected cost estimates.
Infrastructure Visibility	**Spacelift** extracts all your resources enabling discovery and investigatory work ("who owns this resource?")

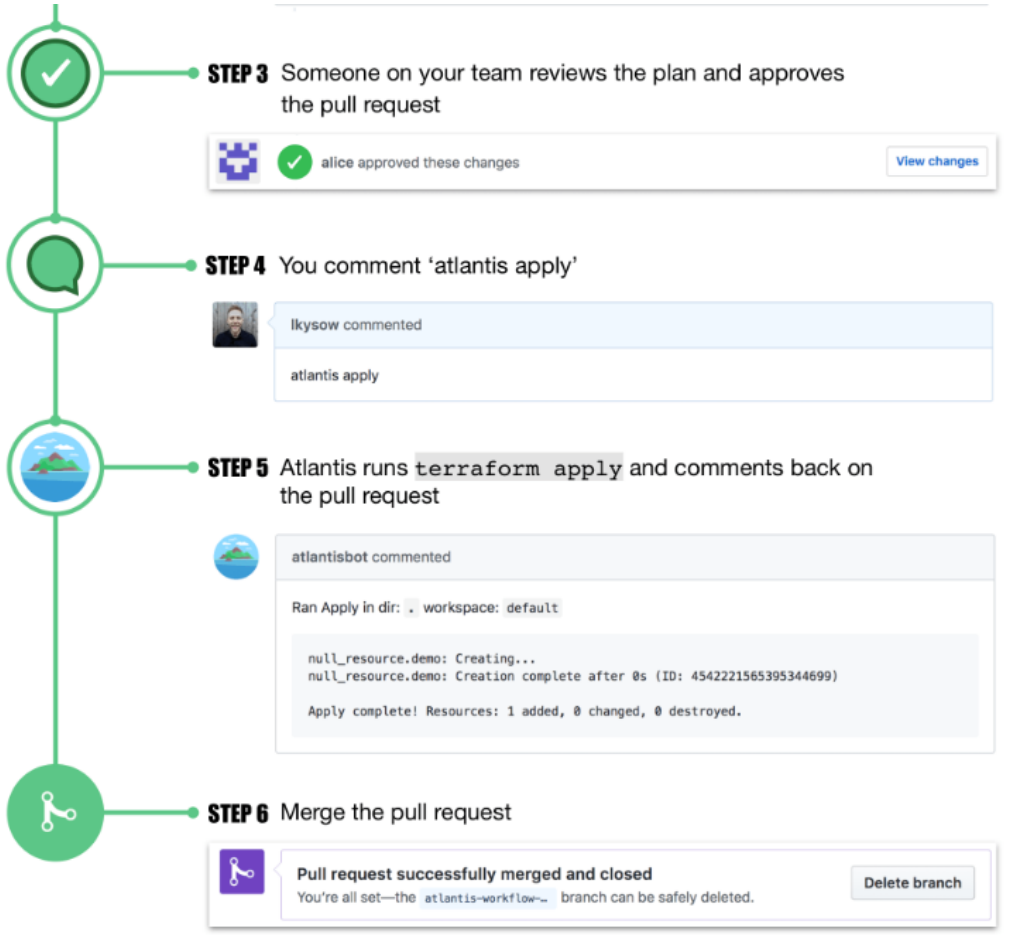
Spacelift

- Blurb: "Spacelift is a sophisticated and compliant infrastructure delivery platform for Terraform, CloudFormation, Pulumi, and Kubernetes"
- Multiple Cloud Providers
- Used and recommended by CloudPosse : Why do you recommend SpaceLift?
  - Drift Detection runs on a customizable schedule surfaces inconsistencies with what's deployed and what's in git.
  - Reconciliation helps you know what's deployed, what's failing, and what's queued.
  - Plan Approvals ensures changes are released when you expect them
  - Policy Driven Framework based on OPA (open source standard) is used to trigger runs and enforce permissions. This is like IAM for GitOps.
  - Terraform Graph Visualization makes it easier to visualize the entire state across components
  - Audit Logs of every change traced back to the commit and filterable by time
  - Affordable alternative to other commercial offerings
  - Works with more than Terraform (e.g. Pulumi)
  - Pull Request Previews show what the proposed changes are before committing them
  - Decoupling of Deploy from Release ensures we can merge to trunk and still control when those changes are propagated to environments
  - Ephemeral Environments (Auto Deployment, Auto Destruction) enables us to bring up infrastructure with terraform and destroy it when it's no longer needed
  - Self-hosted Runners ensure we're in full control over what is executed in our own VPC, with no public endpoints

Atlantis

- Flow

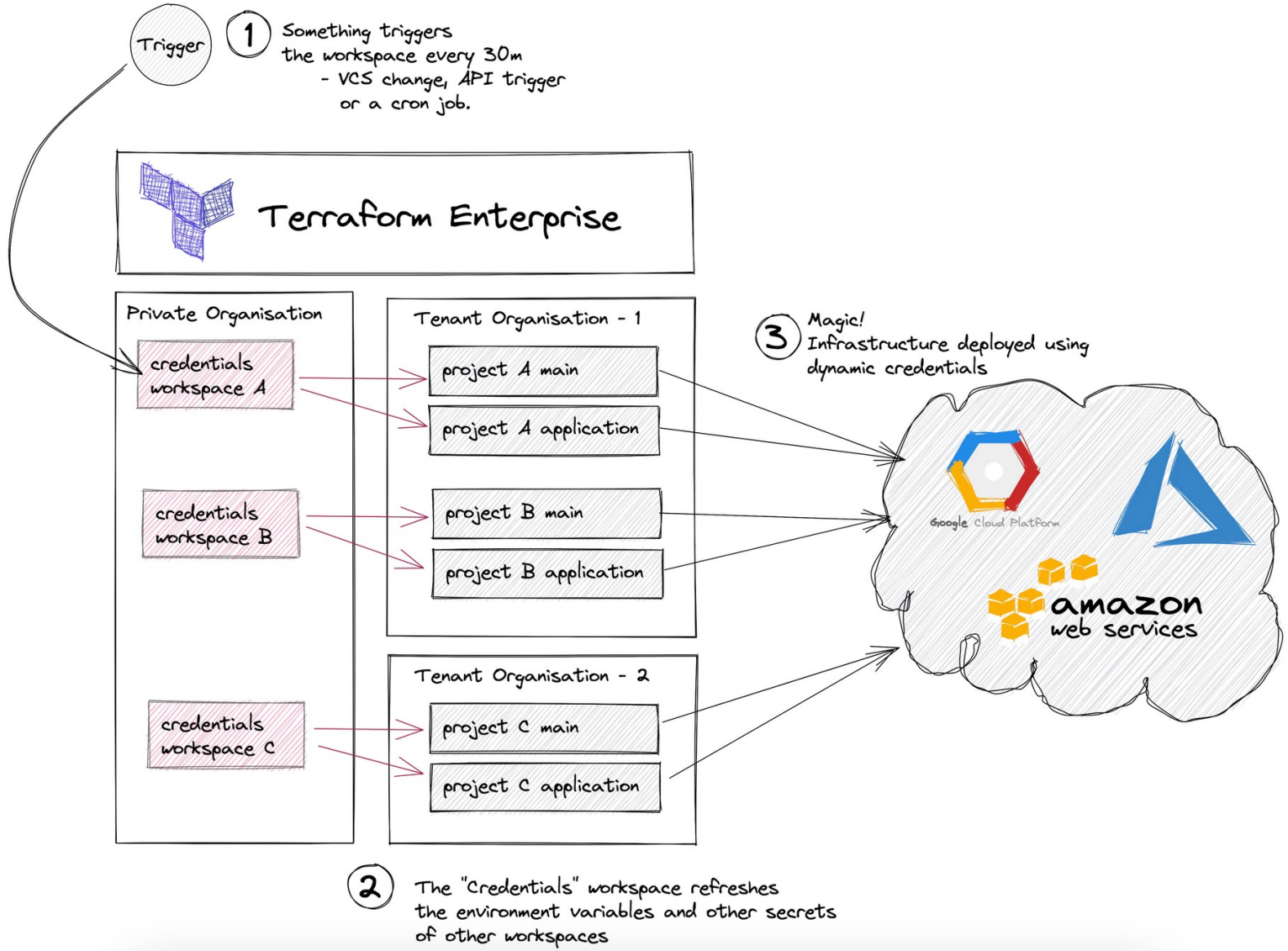




Atlantis – additional info

- Security info from Atlantis
  - Including mitigation proposals
- Case studies / deployments
  - Running Atlantis at Lyft
  - Enforcing best practice on self-serve infrastructure with Terraform, Atlantis and Policy As Code
- Critical review of Atlantis from a competitor (Spacelift - see next slides)
- Terraform AWS module that deploys Atlantis in ECS running Fargate (Spot)

Auxiliary Slides


Trigger
1
Something triggers
the workspace every 30m
- VCS change, API trigger
or a cron job.
Terraform Enterprise
Private Organisation
credentials
workspace A
credentials
workspace B
credentials
workspace C
Tenant Organisation - 1
project A main
project A application
project B main
project B application
Tenant Organisation - 2
project C main
project C application
3
Magic!
Infrastructure deployed using
dynamic credentials
Google Cloud Platform
amazon
web services
2
The "Credentials" workspace refreshes
the environment variables and other secrets
of other workspaces