

Infrastructure in the age of DevOps

AWS & Terraform & Gitlab



Terraform



GitLab

About me

- Name: Traian Ciobanu
- Experience:
 - System administrator
 - Network engineer
 - DevOps Engineer
- Certifications:
 - ✓ AWS Certified SysOps Administrator – Associate
 - ✓ AWS Certified Developer – Associate
 - ✓ AWS Certified Solutions Architect – Associate
 - ✓ HashiCorp Certified: Terraform Associate

How we
deployed
infrastructure
before cloud



How we deployed infrastructure before cloud

- Hardware purchases (often not everything in stock with dealers)
- Prepare rack in DC
- Install equipment
- Configure equipment
- Install & Configure application
- Some level of automation (bash scripts, perl, Ansible, etc)

All this was done manually and human error occurred often

The Cloud



FAST PROVISIONING OF
INFRASTRUCTURE

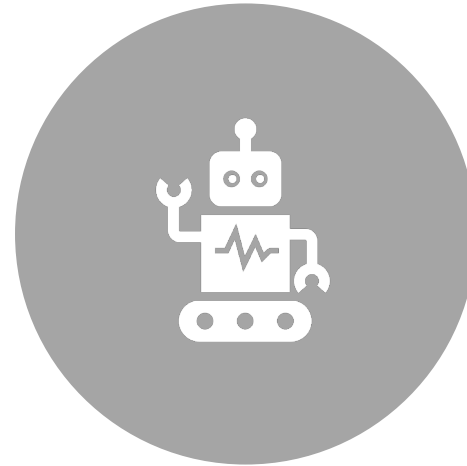


NO UPFRONT COSTS

The Cloud



MANUAL PROVISIONING OF
INFRASTRUCTURE (CLICK OPS)



AUTOMATION
(CLOUDFORMATION)



What is version control and CI/CD

- **Version control**, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time.
- **Continuous integration (CI)** is the practice of merging all developers' working copies to a shared mainline several times a day.
- **Continuous delivery (CD)** is a software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time and, when releasing the software, without doing so manually.

What is IaC

Infrastructure as Code (IaC) is the management of infrastructure (networks, virtual machines, load balancers, and connection topology) in a descriptive model, using the same versioning as DevOps team uses for source code. Like the principle that the same source code generates the same binary, an IaC model generates the same environment every time it is applied. IaC is a key DevOps practice and is used in conjunction with CI&CD.




Terraform

- Terraform is an Open Source Infrastructure as code tool created by HashiCorp.
- Terraform manages external resources (such as public cloud infrastructure, private cloud infrastructure, network appliances, software as a service, and platform as a service) with "providers".
- Users can interact with Terraform providers by declaring resources or by calling data sources.
- Rather than using imperative commands to provision resources, Terraform uses declarative configuration to describe the desired final state.
- Once a user invokes Terraform on a given resource, Terraform will perform CRUD actions on the user's behalf to accomplish the desired state.
- The infrastructure as code can be written as modules, promoting reusability and maintainability.

A large orange circle occupies the left side of the slide, partially cut off by the edge.


GitLab

- **GitLab** is a web-based DevOps lifecycle tool that provides a Git repository manager (code version providing wiki, issue-tracking and continuous integration and deployment pipeline features, using an open-source license, developed by GitLab Inc.
- 
- A yellow dashed line consisting of several curved segments, located in the bottom right corner of the slide.

A large orange circle is positioned on the left side of the slide, partially cut off by the edge.

Cloud meets IaC

With the release of Cloudformation, Terraform, Pulumi, and other tools we enter in the age of infrastructure as code and principles that applied to software development can be applied to infrastructure

A series of four yellow dashed line segments are arranged in a curved, upward-pointing arc in the bottom right corner of the slide.



Challenges on Infrastructure as Code:

#1. Not using Git


- Create and test code locally
- Apply code directly from local

#2. Using Git the wrong way

- No collaboration
- No pull/merge requests
- No code review
- No tests

#3 Accessing infrastructure directly to apply the changes

Organizations provide direct access to engineers to K8S clusters, AWS, Azure, GCP cloud platforms and that makes hard to trace who executed what.



A large orange circle is positioned on the left side of the slide, partially cut off by the edge.


GitOps – Doing “Infrastructure as Code” right

“GitOps” concept born in 2017 to treat
“Infrastructure as code” the same way as
“Application as Code”. GitOps eliminates the
flaws of “IaC”.






GitOps – Doing “Infrastructure as Code” right

- Store the code in dedicated “IaC” repository. This enables version control and improves collaboration.
 - Use pull/merge request for collaboration and approval process. This brings an end to end automated process and more transparency on the deployment and management. Due to multiple reviews and testing, the quality of “IaC” improves a lot.
 - Continuous deployment model. In GitOps, there are two way to apply the changes to the infrastructure. This eliminates direct access to the infrastructure.
- 



GitOps – Doing “Infrastructure as Code” right

- **Push deployment** - CI/CD servers trigger the job which applies the changes to the infrastructure. This can be done using git actions or manual triggers with the pipeline job.
 - **Pull deployment** - in Pull deployment mode, agents running on the source infrastructure which always looks for the desired state from the git repository. In Kubernetes, an agent will be installed on the Kubernetes cluster which regularly monitors the repository. This agent compares the desired state (Git repository) with the actual state (Kubernetes cluster environment) frequently. If there are any changes in the desired state, it pulls the configuration and applies it.
- 

Why choose Terraform and GitLab



EASY AND HUMAN
READABLE



FAST START

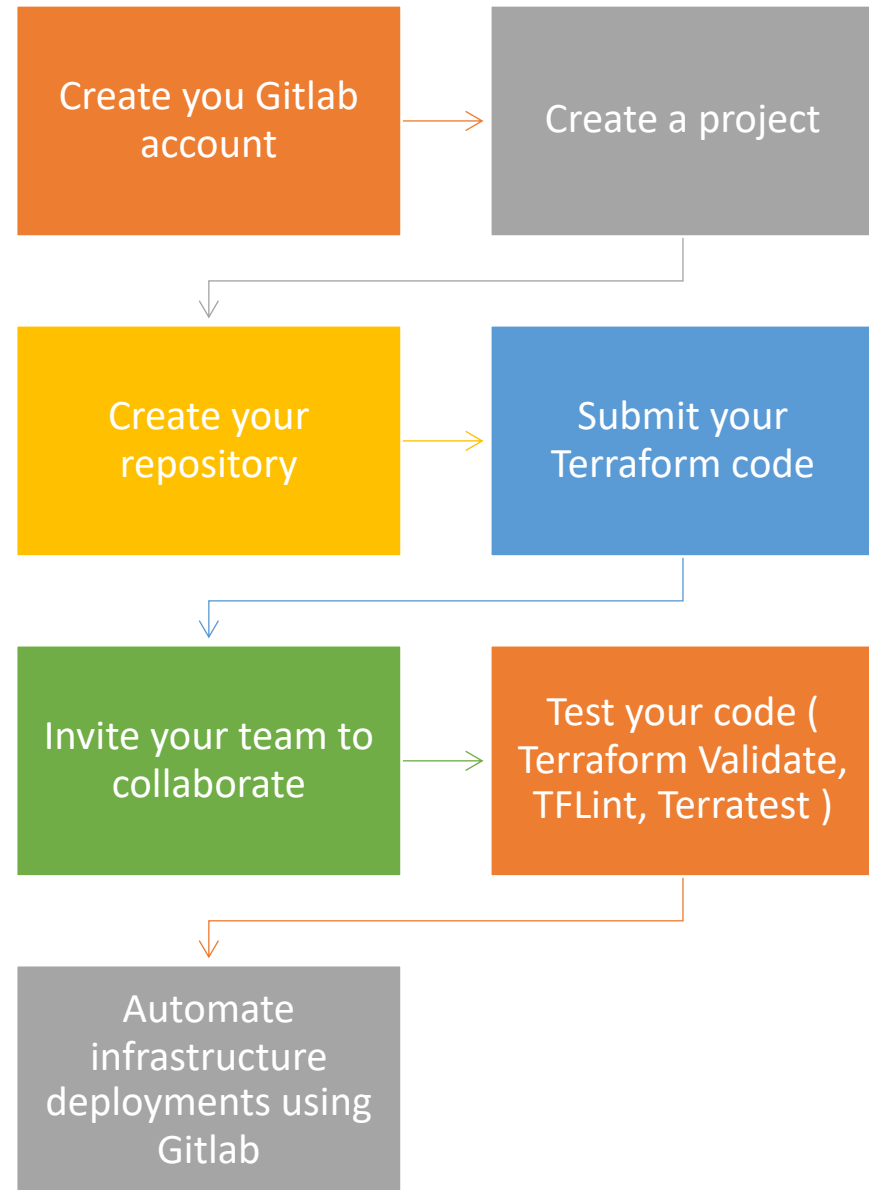


DOCUMENTATION



WIDE
COMMUNITY

Using Gitlab and Terraform for the right IaC way





Demo Time

Deploy simple AWS Infrastructure using Gitlab pipeline