Troubleshooting Terraform

Ned Bellavance
FOUNDER, NED IN THE CLOUD LLC

@ned1313 www.nedinthecloud.com

Overview

Validating configurations

Enable verbose logging

Resource taints

Crash logs

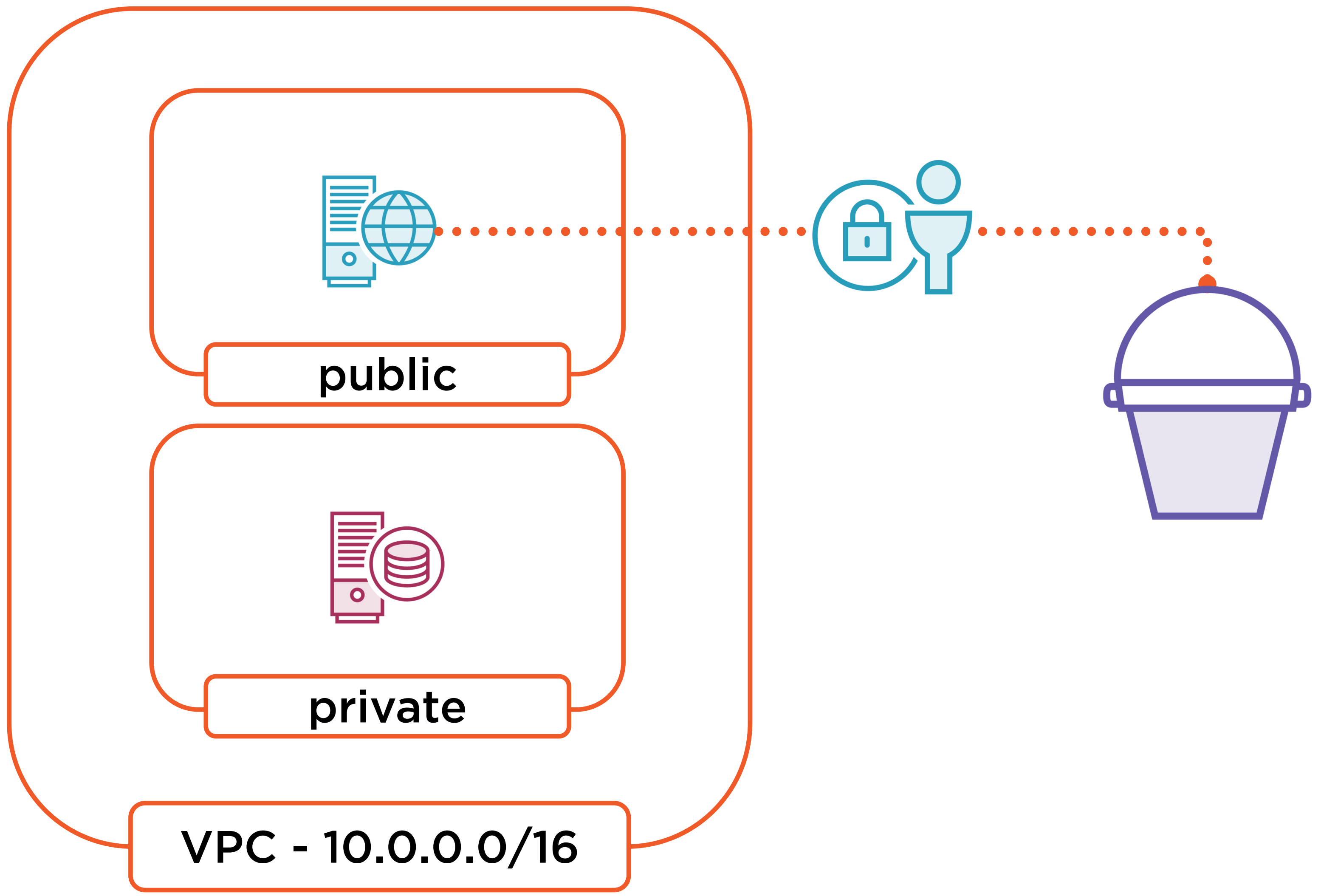
Globomantics Environment

Application team update

Troubleshooting deployment issues

Panic! at the Terraform

Application Update



Types of Errors

Command error

Syntax validation

Provider validation

Deployment error

Panic!

Command Error

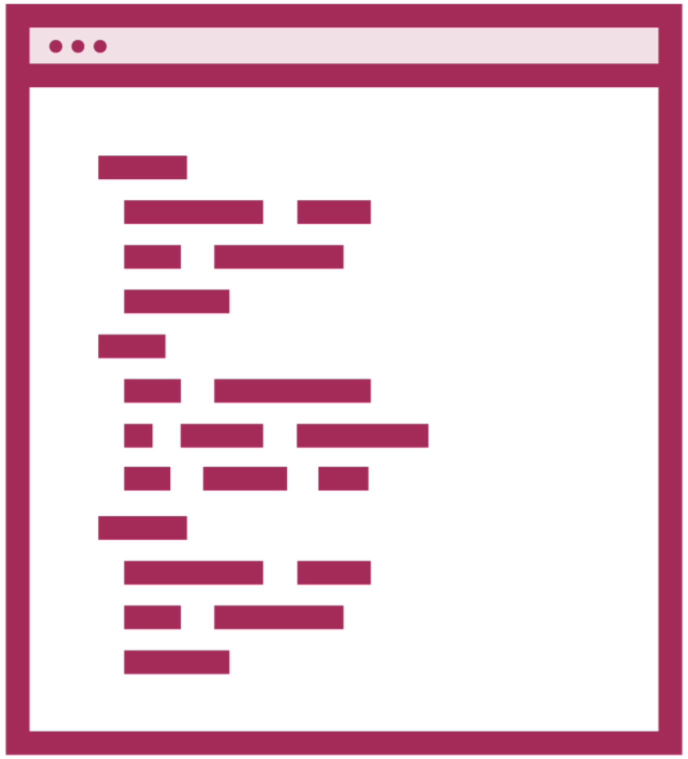
Happens at the command line

Bad CLI syntax or arguments

Use the help argument

Read the docs

Syntax Validation



Terraform init first

Checks syntax and logic

Does not check state

Manual or automatic

Automation checks

Provider Validation and Deployment Errors

Happens during plan or apply

Read the error message, then read it again

Enable logging for more detail

Use taint to destroy bad resources

Verbose Logging

Exposes Terraform actions

Enabled through TF_LOG

Write to file with TF_LOG_PATH

Trace is most verbose

Useful in automation

Resource Taints

Marks a resource for recreation

Terraform will taint automatically

Identify resources by address

Resources can be untainted as well

Taint and Untaint Command

```
# Command syntax
terraform taint [options] address

# Example single resource
terraform taint aws_instance.example

# Example module or collection
terraform taint aws_instance.collection[0]
terraform taint module.asg.aws_instance.example

# Untaint syntax
terraform untaint [options] address

# Example single resource
terraform untaint aws_instance.example
```

Crash Log

Generated when Terraform panics

Caused by Terraform or provider

Similar to trace logging

Open an issue on GitHub

Summary

Errors can and will happen

Read the error twice

Enable logs

Use taints

Coming up:

- Integrating Terraform with Jenkins
- Enabling automation for Terraform