



# Infrastructure Testing

The Road to Reliability

11.03.2020, DevOps Gathering 2020

Constantin Weïer |  iSibnZe



Infrastructure & Automation Lead at Novatec  
Consultant, Trainer, Traveller, Music Addict



# Quick show of hands ...





puppet



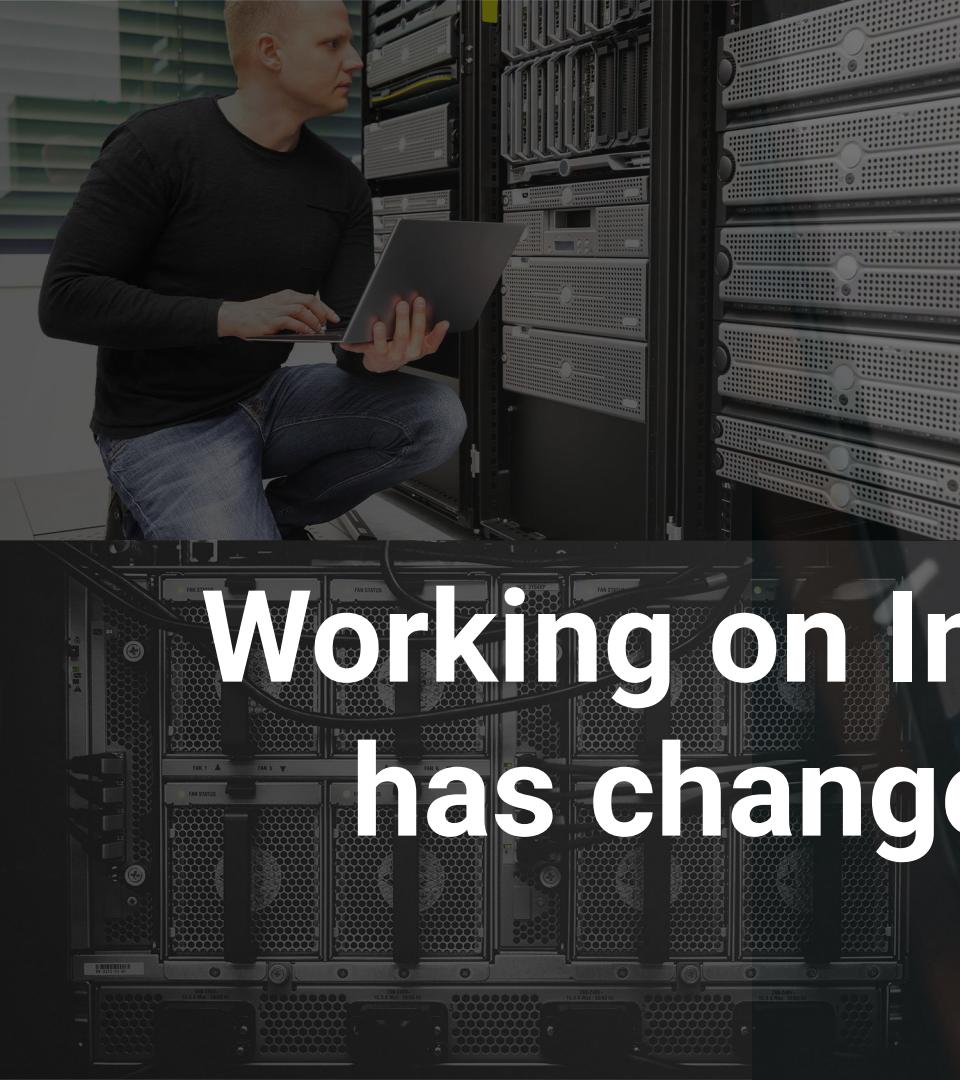
2010



HashiCorp



2020

A black and white photograph showing a man from the side, sitting on a step ladder and working on a large server rack. He is holding a laptop and looking at it while reaching into the server unit. The server rack is filled with various components and drives. In the foreground, there is another server unit with multiple drive bays and fan grilles.

# Working on Infrastructure has changed ... a lot!



# We moved the responsibility ...

---

... but our **engineering** lags behind

- Version Control
- Pipelines
- Transparency
- Steady environment
- **Testing**

# Why?



# Yes, why?

---

- **Why not!**
  - Why do you test your application code?
  - Why should infrastructure code be different?
- **Logic:** If the infrastructure code contains any sort of logic
- **Verify:** The infrastructure behaves as (we think) we specified it in code
- **Validate**
  - Codifying expected behaviour triggers thinking about it
  - Are we implementing the right thing?
- **Invariants:** Make sure hard requirements always hold true
- **Robustness** against changes (uncover interdependence)

## Testing is important for automation!

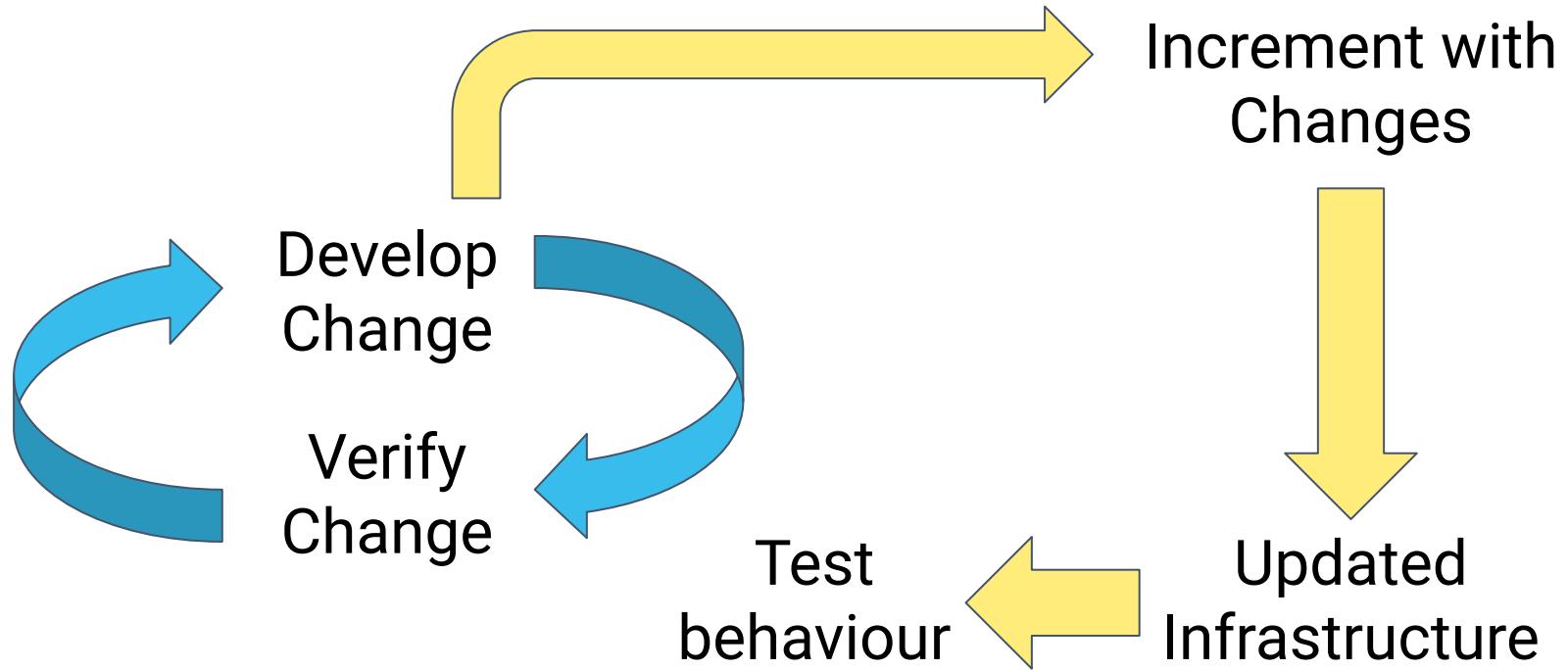
A dark, moody photograph of a turbulent river flowing through a mountainous valley under a cloudy sky. The water is white and foamy, indicating strong currents. The background shows steep mountains covered in green vegetation and patches of snow or ice. The overall atmosphere is dramatic and powerful.

# What is your Workflow? ~



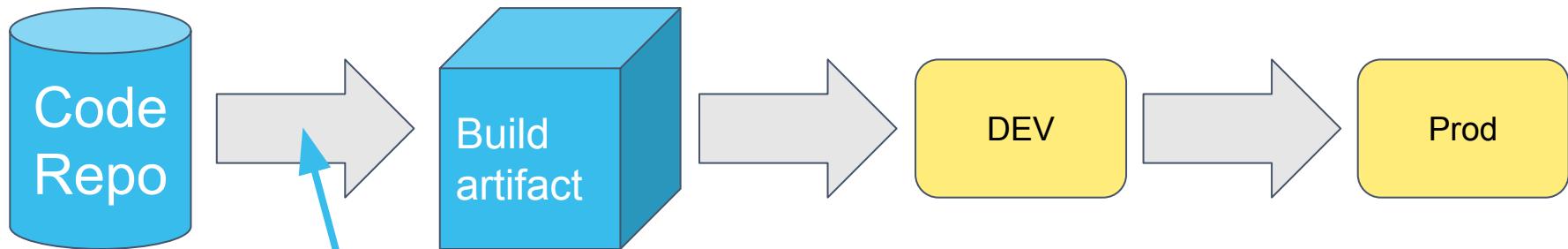
# The Development Cycle

---



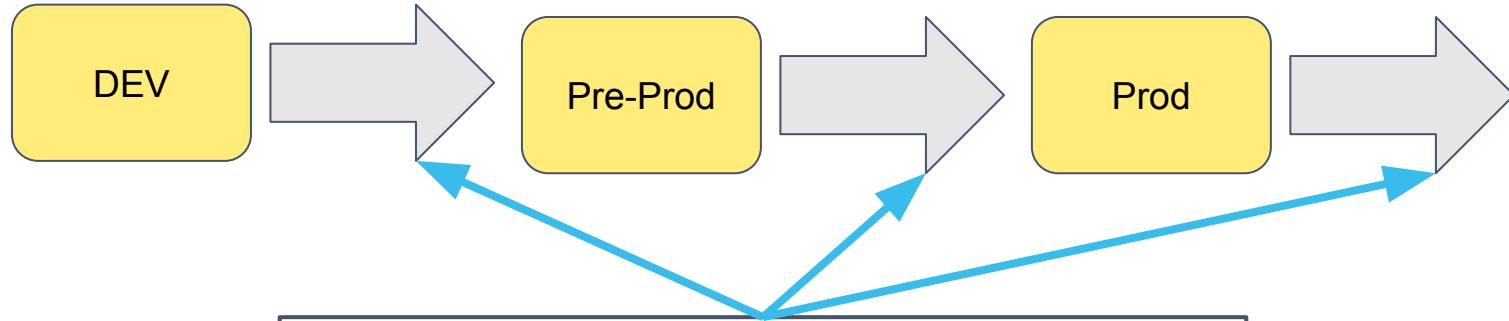
# In the Build Pipeline

---



1. Build and test infrastructure code
2. Failing tests won't produce a usable artifact

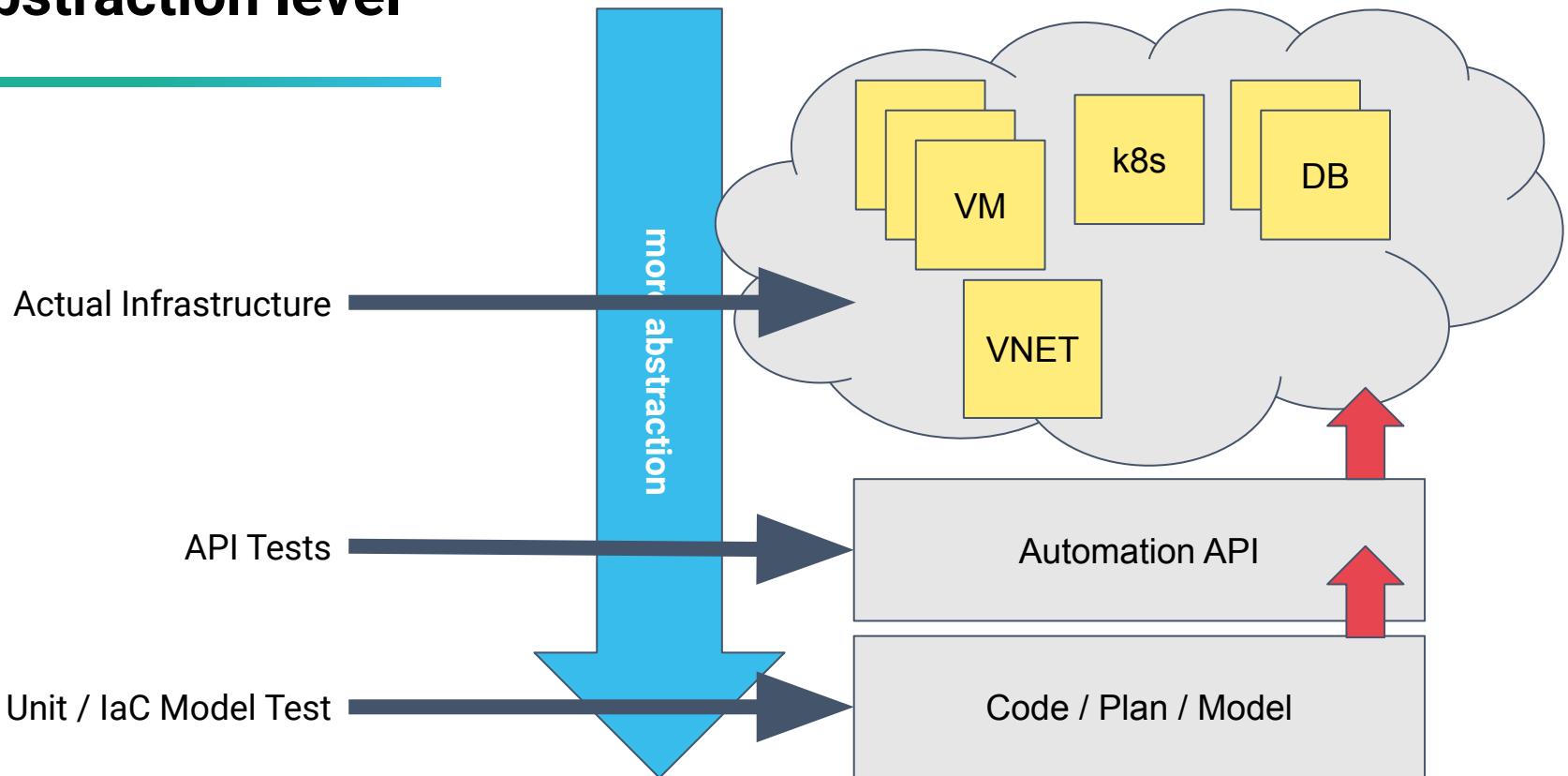
# In the Deployment Pipeline



1. Provision Infrastructure
2. Run tests
3. Failing tests fail the pipeline

# Abstraction Level?

# Abstraction level



# Test Abstraction Mnemonics

---

- **Model tests:** I don't trust my code
- **API tests:** I don't trust my tool (nor my code)
- **Real infrastructure tests:** I have trust issues :)

**Real infrastructure tests are expensive, but the only way to know for sure ...**



NOVATEC

БЭЙСИК АВК НЦ

Нужны вам расширенные функции?!

Высокоскоростные устройства?!

Установка внешней функции?!

ОЗУ ?48

ЖДУ

5 LET A=1.0000001

10 LET B=A

15 FOR I=1 TO 100

20 LET A=A\*A

25 LET B=B^2

30 NEXT I

35 PRINT A,B,"WWW.LENINGRAD.SU/MUSEUM"

RUN

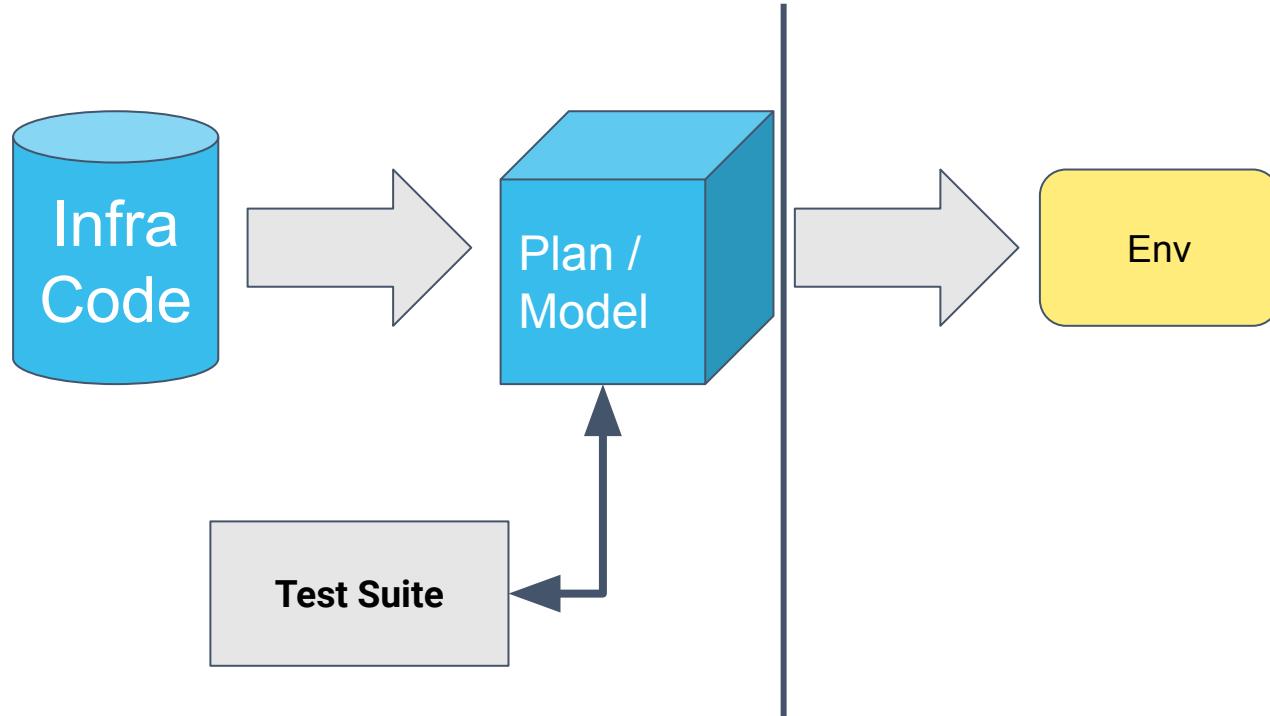
568044

1202420

WWW.LENINGRAD.SU/MUSEUM

ОСТ СТРОКЕ  
ЖДУ

35



Model testing is **fast** and **cheap** and  
helps detecting **faulty logic** and  
**illegal inputs!**

# Pulumi Model Testing

---

The example shown in the presentation can be found here:

<https://github.com/i7c/infratests>

# And what about Terraform?

---

You will have to apply some hacks to do model testing here. You can get your hands on a terraform plan:

```
terraform plan -no-color -out "tf.plan"  
terraform show -json "tf.plan"
```

# Infrastructure under Test

Testing Real Infrastructure

# Classification of IuT Pieces

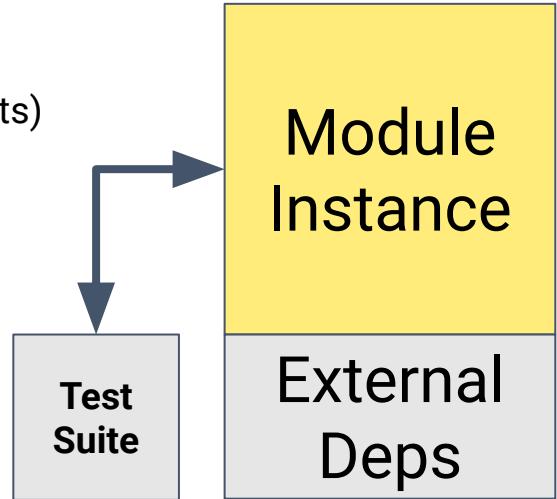
---

- Dedicated ↔ Reused
- Ephemeral ↔ Persistent
- Immutable ↔ Mutable
- Testing ↔ Productive

# Example 1: Dedicated, Ephemeral, Immutable, Testing

---

- Test **an abstraction** in the Infrastructure Code Base (Module Tests)
- Test is comprised of
  - required external resources
  - module instantiation
- For every test execution: **Create, Test, Destroy**



**Tests run off site and assert that the module works as expected.**

# Example 1: Dedicated, Ephemeral, Immutable, Testing

---

## Pros

- Isolated
- No interference with prod. infrastructure
- Cheap compared to persistent IuT

## Cons

- High execution times / slow iterations
- Only tests the abstraction
- Glue code might be extensive
- Modules are not always good for starting

# JUnit Test Suite

---

The example shown in the presentation can be found here:

<https://github.com/i7c/infratests>

## Example 1b: Dedicated, Persistent, Mutable, Testing (3)

---

- Keep test infrastructure around
- Provision updates, not from scratch
- Faster
- Higher quality, if prod is also mutable!

# Example 2: Reused, Persistent, (Im)Mutable, Productive

---

- Test **Productive** infrastructure as it changes
- Can enhance a pipeline (**stages**)
  - Test rollouts (automation)
  - **Safety net** for High Availability
- **Mutable**: Provision infrastructure then check
- **Immutable**: Provision infrastructure, check, then switch

**Keep in mind: Tests may interfere with your production workload!**

# How to find good test cases?

---

- Test things your team has **broken** before!
- Test **updates**!
- Test **critical paths** for availability!
- Test **contracts**!

# Thank you for being here! Questions? Discussions?

---

Feel free to reach out:

**Constantin Weißen | @iSibnZe**

[constantin.weisser@novatec-gmbh.de](mailto:constantin.weisser@novatec-gmbh.de)

# References

---

<https://github.com/i7c/infratests>

<https://www.pulumi.com/blog/testing-your-infrastructure-as-code-with-pulumi/>

<https://github.com/gruntwork-io/terratest>

<https://www.inspec.io/>

<https://github.com/hashicorp/terraform-config-inspect>

Third-party images:

- [https://upload.wikimedia.org/wikipedia/commons/1/1e/Highway\\_401\\_by\\_401-DVP.jpg](https://upload.wikimedia.org/wikipedia/commons/1/1e/Highway_401_by_401-DVP.jpg)
- [https://upload.wikimedia.org/wikipedia/commons/d/d8/Grand\\_Coulee\\_Dam\\_spillway.jpg](https://upload.wikimedia.org/wikipedia/commons/d/d8/Grand_Coulee_Dam_spillway.jpg)