



IaC and GitOps for DevSecOps

Key Takeaways

All rights reserved to nnSoftware GmbH

No part of this publication may be reproduced, copied, transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of nnSoftware GmbH

About TechWorld with Nana

TechWorld with Nana is an established name in the DevOps and Cloud industry, and it stands for the quality trainings helping 1,000s of engineers acquire the most in-demand skills in this field.



Our mission is enable individual engineers as well as companies to take advantage of the recent developments in Cloud and DevOps fields, to use technologies and concepts in order to create efficient, automated, streamlined DevSecOps processes in organisations.



Infrastructure as Code for Security

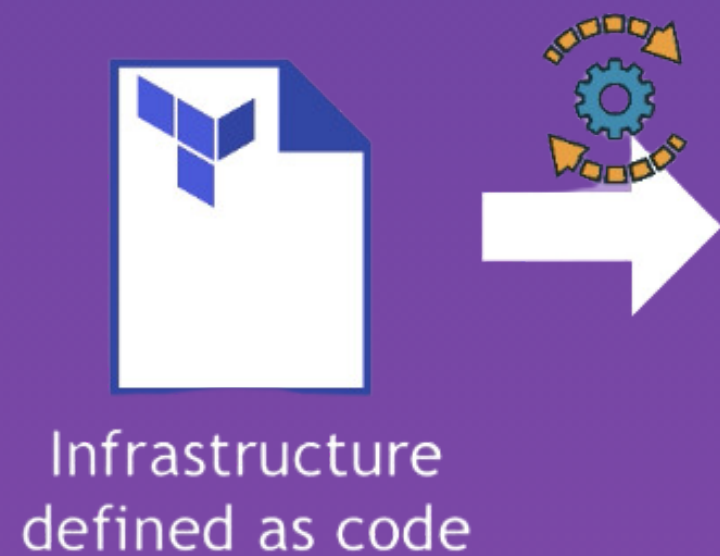
IaC for security



Without IaC

- ❌ Slow manual configuration
- ❌ Prone to human errors
- ❌ Inflexible, **hard to replicate** across environments
- ❌ Dependent on well written and up-to-date documentation

IaC for security

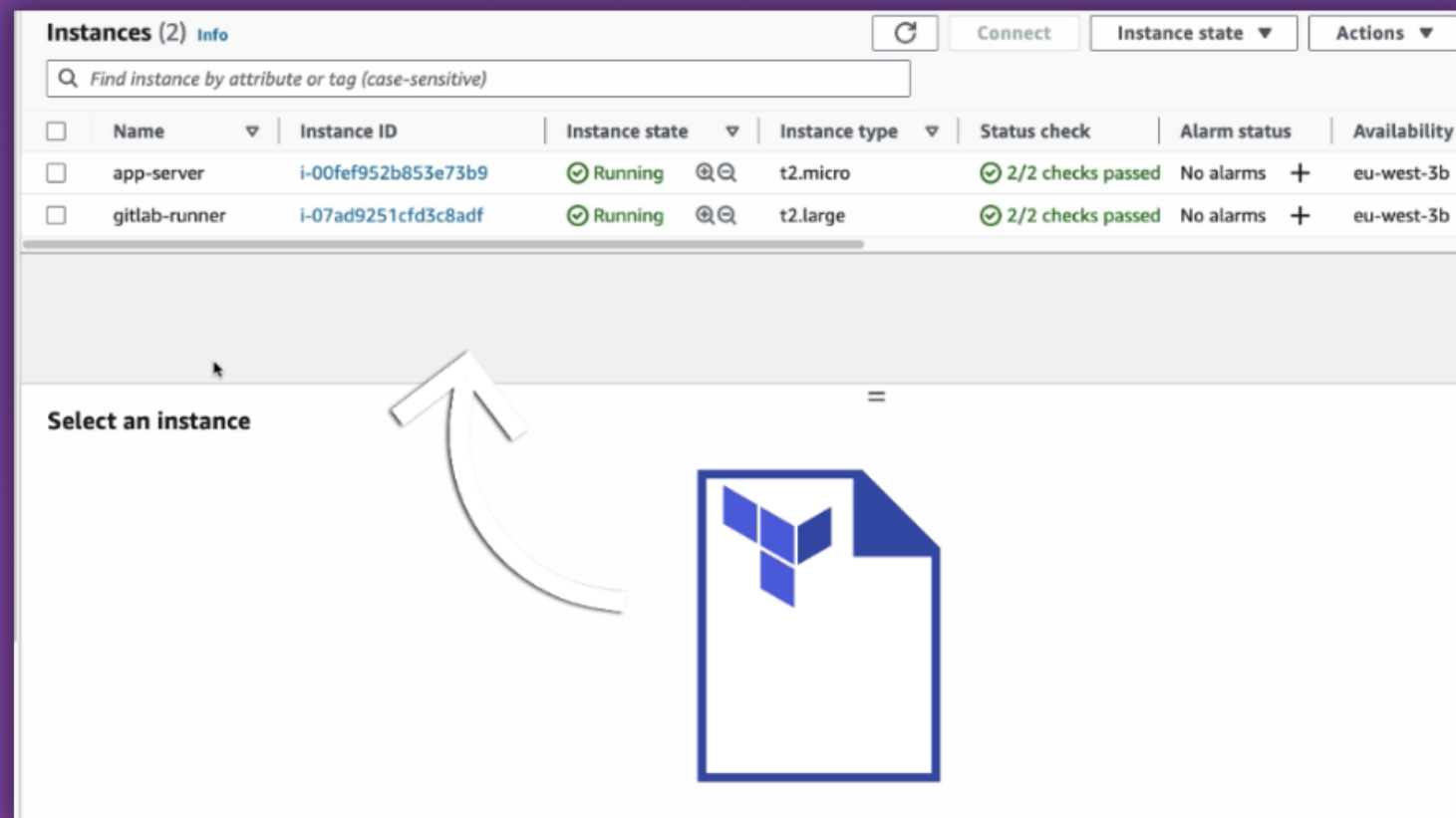


Some Benefits of IaC

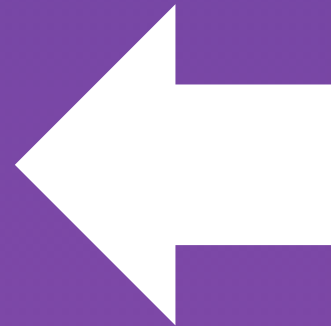
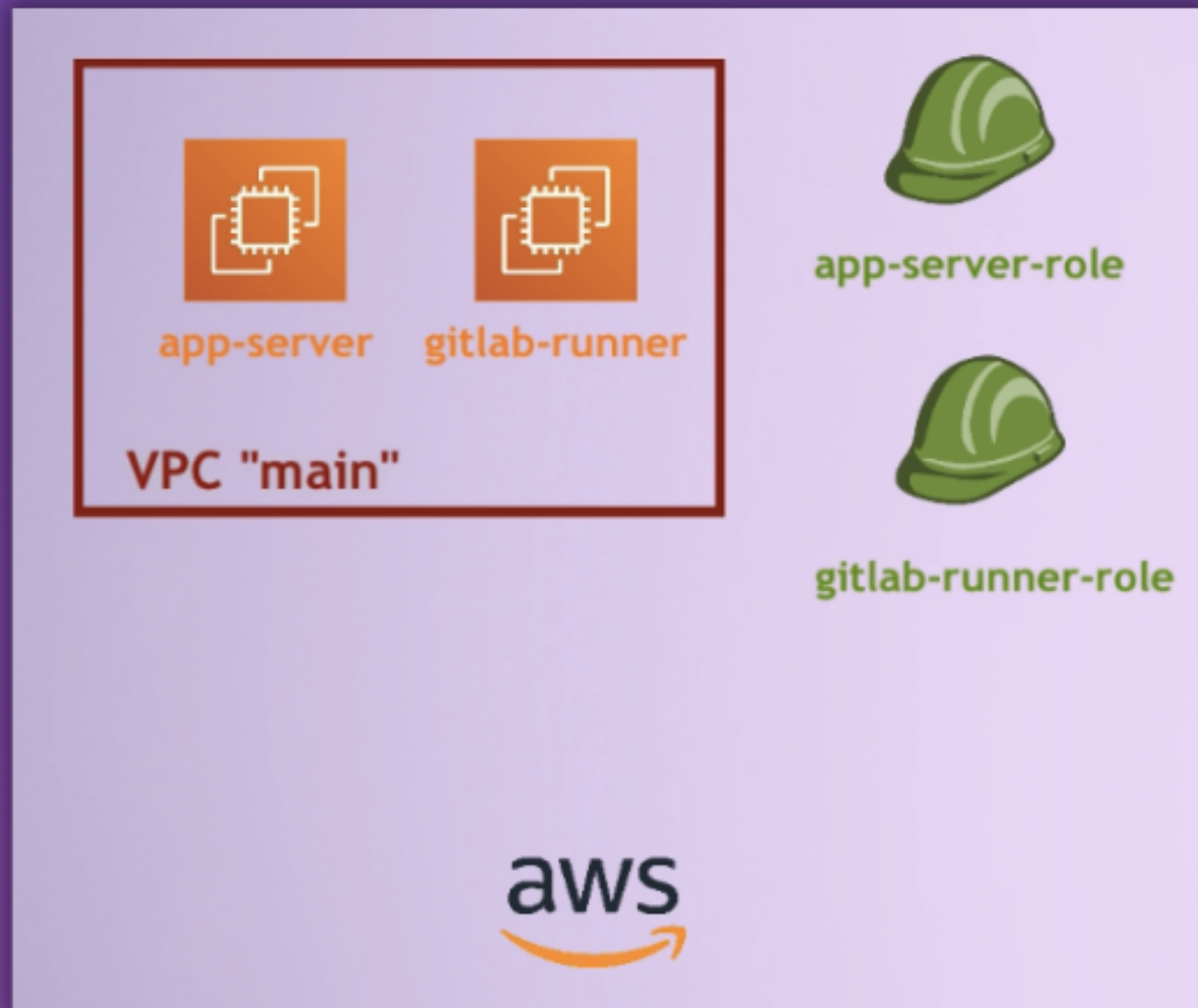
- ✓ Efficient
- ✓ Consistency and Reproducibility

Security Benefits

- ✓ Increased transparency, because code itself is documentation
- ✓ We can use tools to validate the code for security misconfigurations
- ✓ Code can be reviewed, tested and shared easily



Infrastructure created by IaC



```
EXPLORER
├── TF-AUTOMATION
│   ├── .terraform
│   ├── scripts
│   │   ├── script-gitlab.tpl
│   │   └── script.tpl
│   ├── .gitignore
│   ├── .terraform.lock.hcl
│   ├── data.tf
│   ├── locals.tf
│   └── main.tf
├── providers.tf
├── README.md
├── terraform.tfvars
└── variables.tf

main.tf
191 module "ec2_app_server" {
192     depends_on = [aws_security_group.app-server]
193     # TF module that creates EC2 instances: https://registry.terraform.io/modules/t
194     source      = "terraform-aws-modules/ec2-instance/aws"
195     version     = "5.2.1"
196
197     name = "app-server"
198
199     instance_type           = "t3.small"
200     availability_zone       = element(data.aws_availability_zones.available.names, 0)
201     ami                    = data.aws_ami.ubuntu.id
202     iam_instance_profile    = data.aws_iam_instance_profile.app-server-role.name
203     associate_public_ip_address = true
204     vpc_security_group_ids  = [aws_security_group.app-server.id]
205     subnet_id              = module.vpc.public_subnets[0]
206     user_data              = base64encode(local.script)
207
208     tags = {
209         Terraform = "true"
210         Environment = var.env_prefix
211         Name       = "app-server"
212     }
213 }
```



GitOps - DevOps for IaC

GitOps - DevOps for IaC



What is GitOps?

- Takes best practices of application development, such as version control, collaboration and CI/CD and applies them to infrastructure

Key Concepts and Principles of GitOps

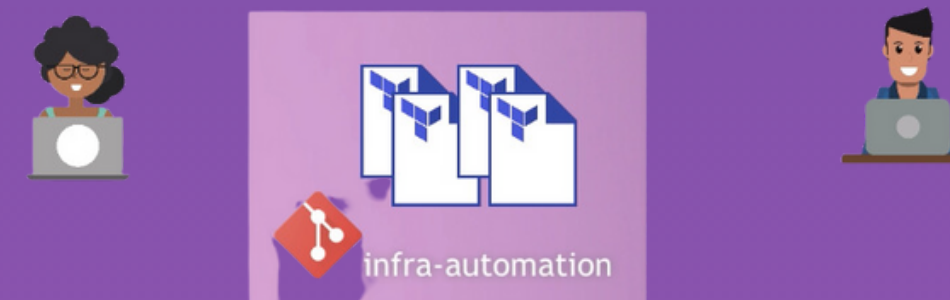
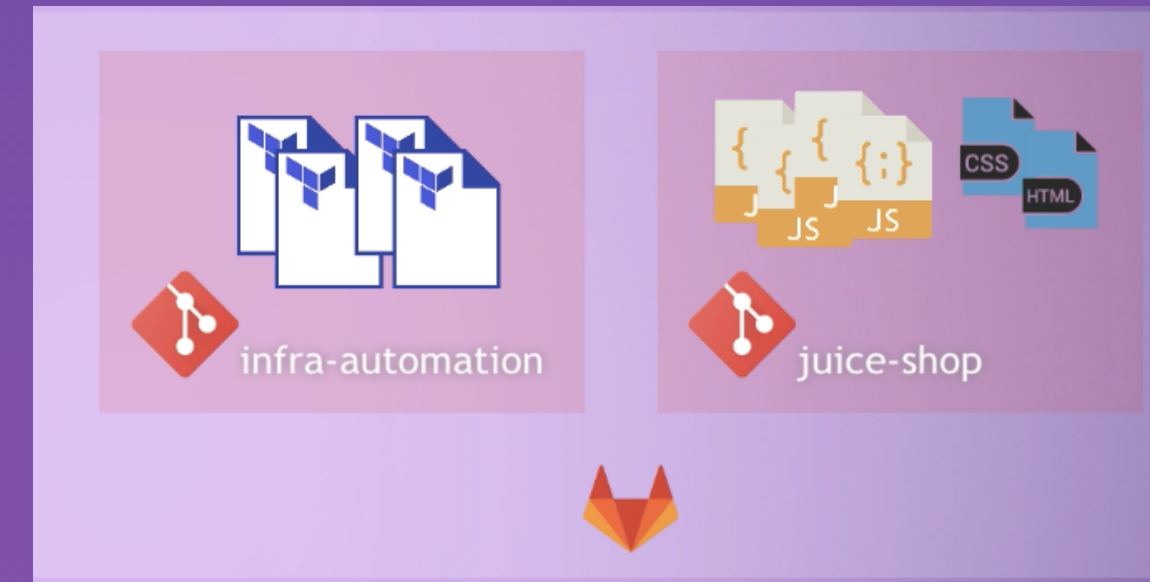
- Infrastructure as Code
- Git as the single source of truth
- Collaboration
- Automation: CI/CD pipeline to automate process of syncing the actual state of the system with the desired state defined in the Git repository



GitOps - DevOps for IaC

Version Control with Git

- Git naturally becomes part of IaC
- IaC code can be versioned, just like application code
- Enables you to track changes, roll back to previous configurations if needed
- Use Git repository features like code reviews, merge requests etc.
- Work on infra configurations simultaneously



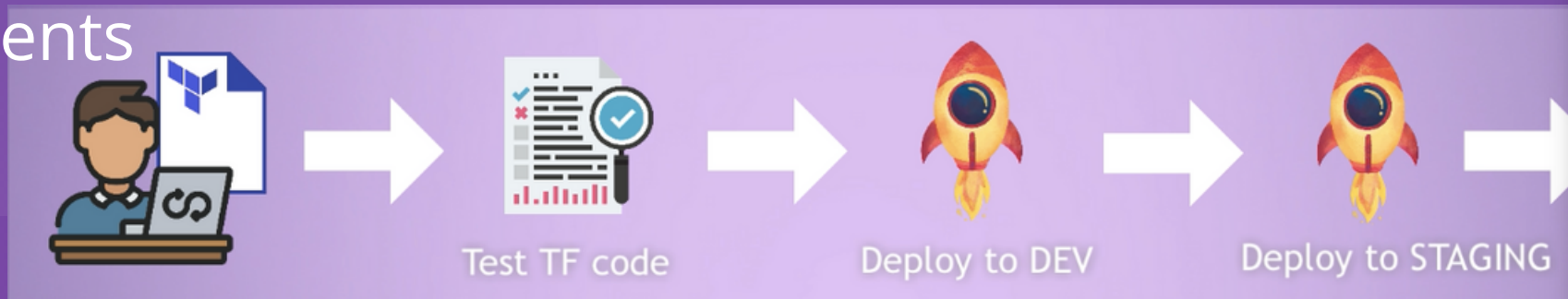
Collaborate on IaC code



GitOps - DevOps for IaC

CI/CD for IaC

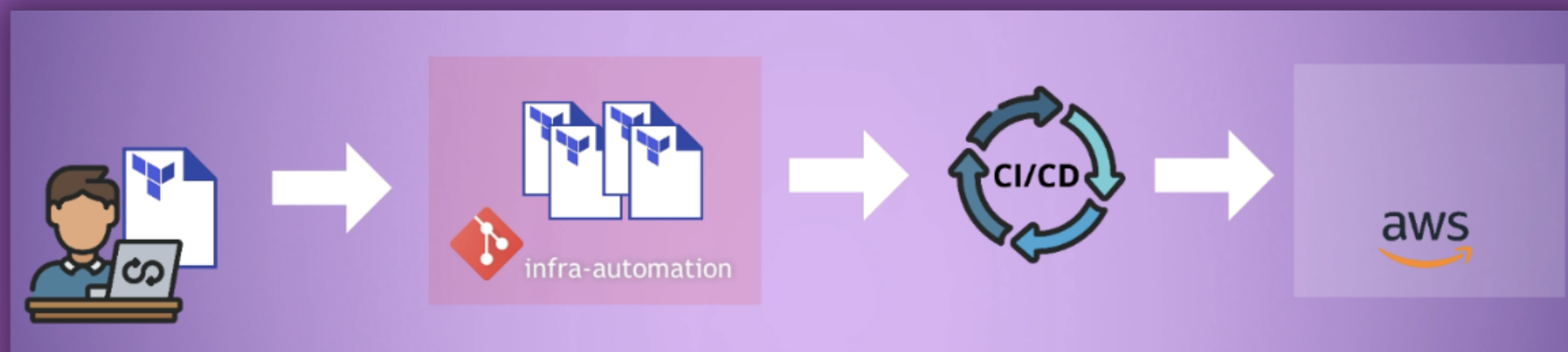
- Automate testing and deployment of infrastructure changes
- Multi-stage deployments



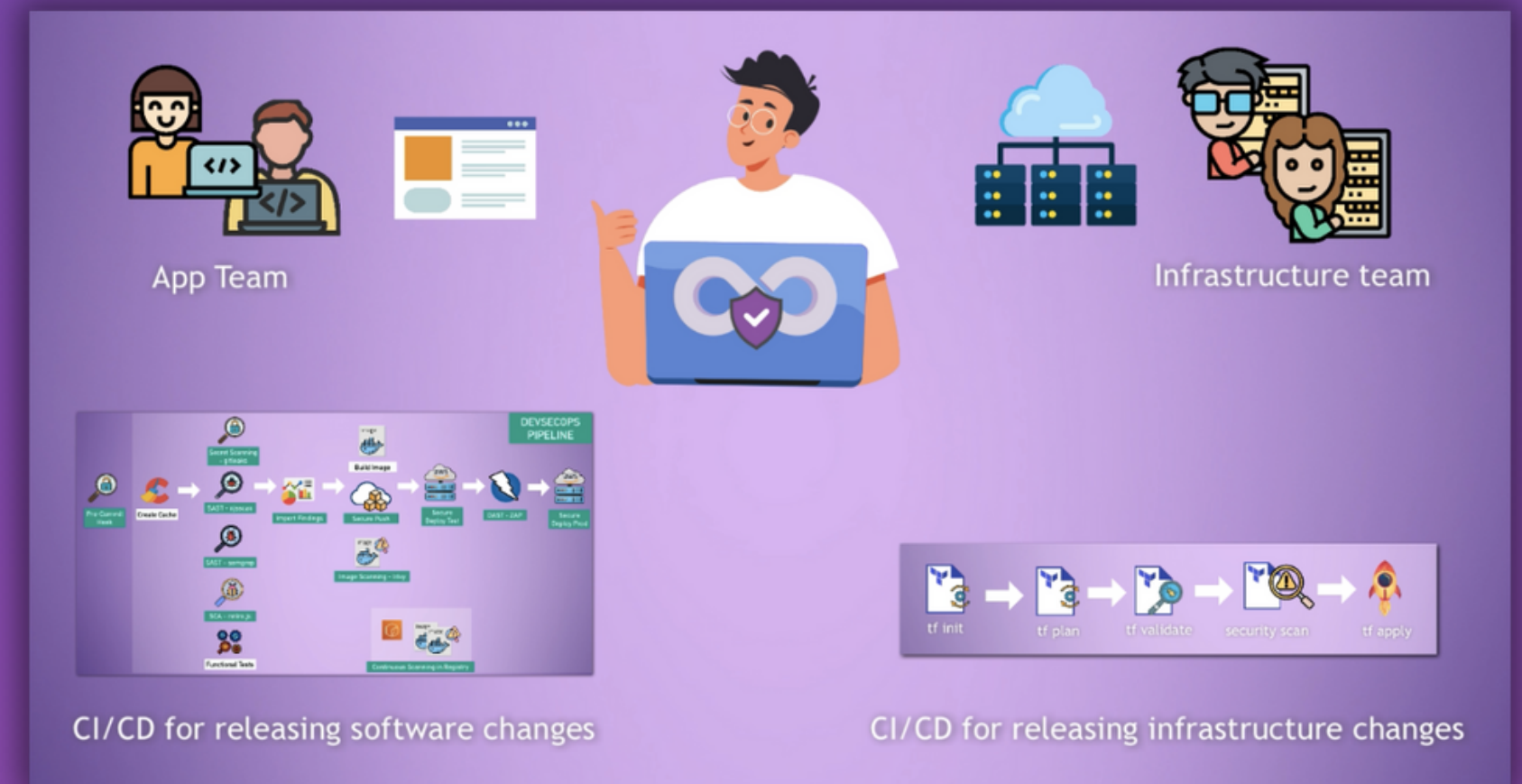
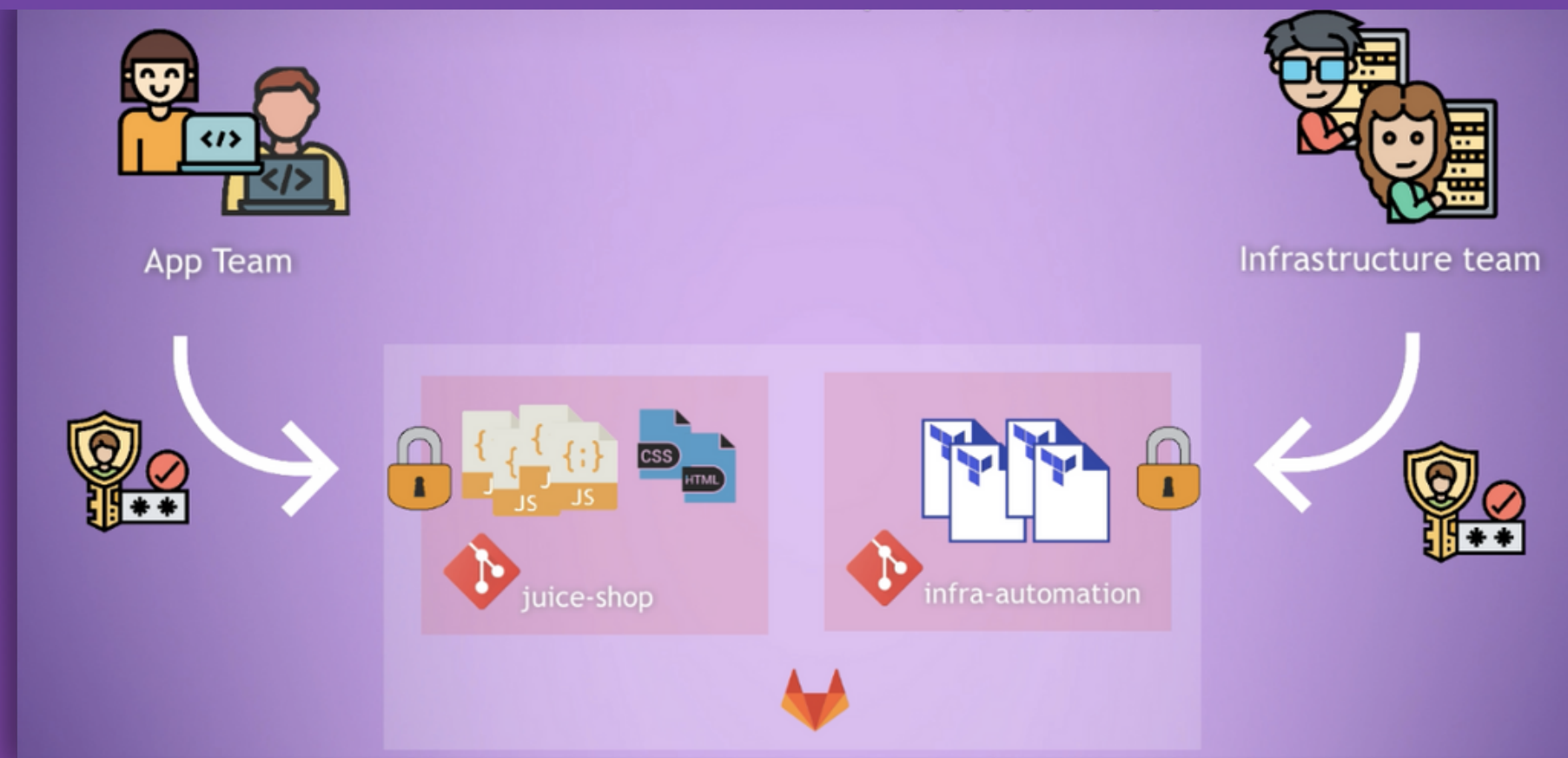
```
.gitlab-ci.yml
20  script:
21    - terraform init
22  artifacts:
23    paths:
24      - .terraform/
25      - .terraform.lock.hcl
26
27  build:
28    stage: build
29    script:
30      - terraform plan -out "planfile"
31  artifacts:
32    paths:
33      - planfile
34
35  deploy:
36    stage: deploy
37    script:
38      - terraform apply -input=false "planfile"
39    when: manual
```

Git becomes the single source of truth

- Having CI/CD configured, every infrastructure change will be **applied automatically** to the infrastructure
- This means just by looking at Git repository code, we know the current infrastructure state



DevOps for App code & Infra code



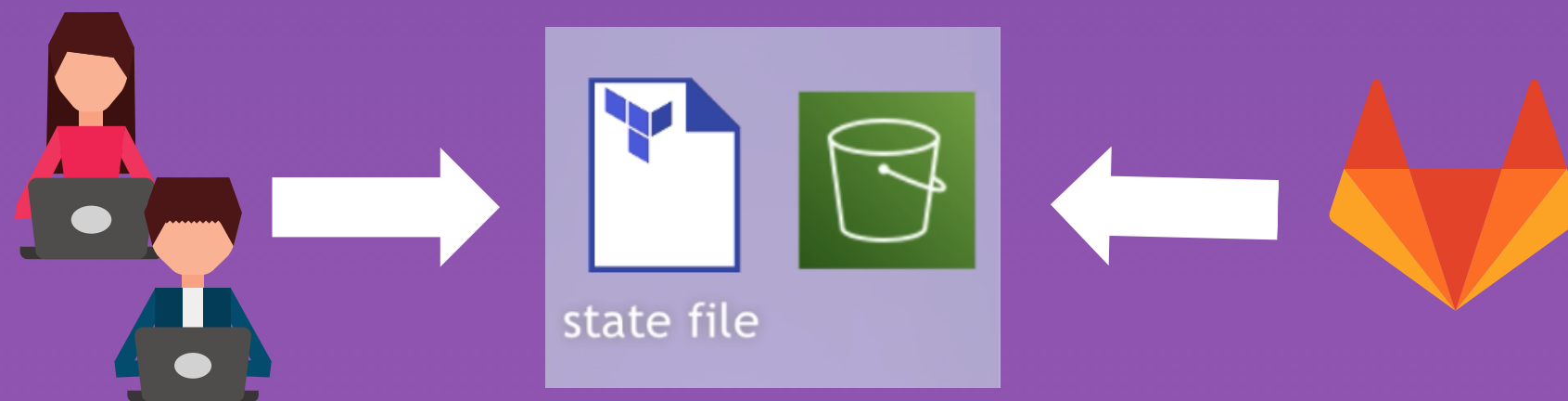
Terraform state

What is TF State?

- TF must store state about your managed infrastructure and configuration
- This state is used by TF to map real world resources to the configuration

Best Practice - Configure remote state

- Central storage for infrastructure state
- Remote store, like S3 bucket, instead of local state on local filesystem



```
.gitlab-ci.yml U providers.tf M x
providers.tf > terraform > backend "s3" > bucket
1 terraform {
2   backend "s3" {
3     bucket = "infra-bucket"
4     key = "infra/state.tfstate"
5     region = "eu-west-3"
6   }
7   required_providers {
8     aws = {
9       source = "hashicorp/aws"
10      version = "~> 5.3"
11    }
12  }
13 }
14
```


Automated Terraform security scan

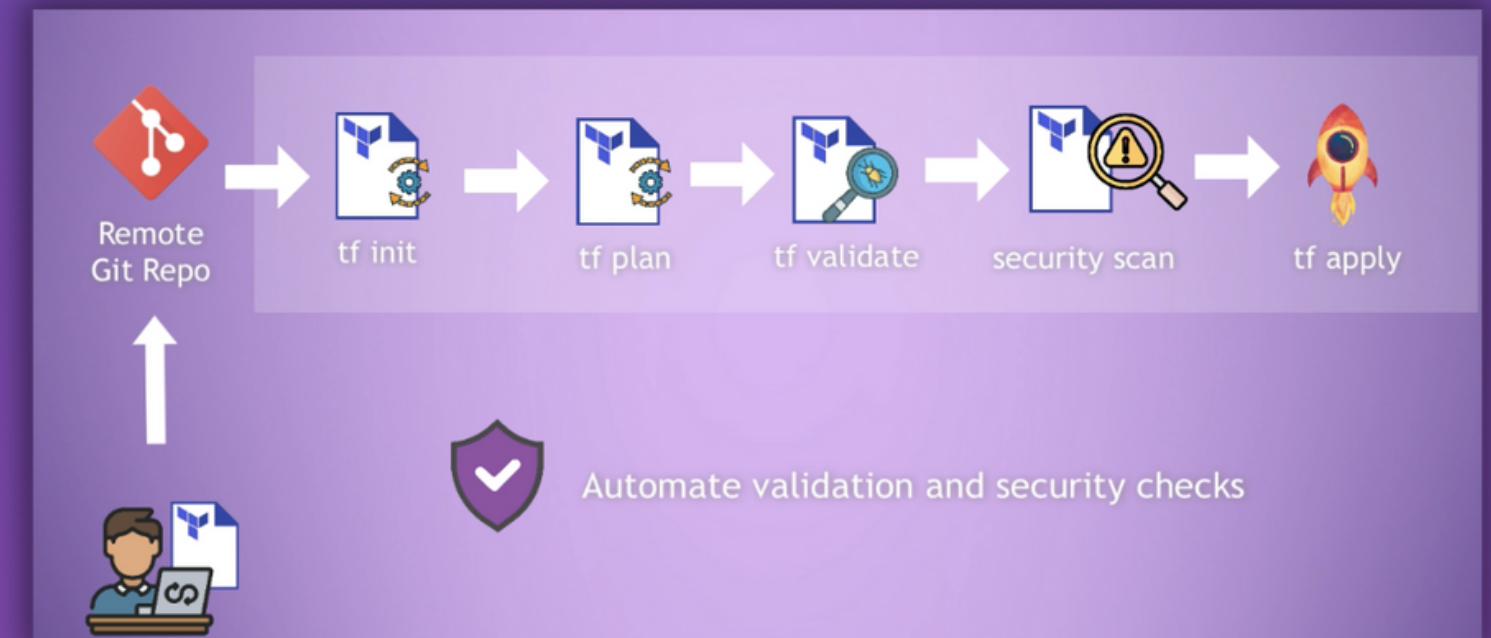
- Turn CI/CD pipeline into DevSecOps pipeline, scanning our IaC for security misconfiguration

"terraform validate" command

- Check script for syntax validity, general correctness of attributes, variables, modules

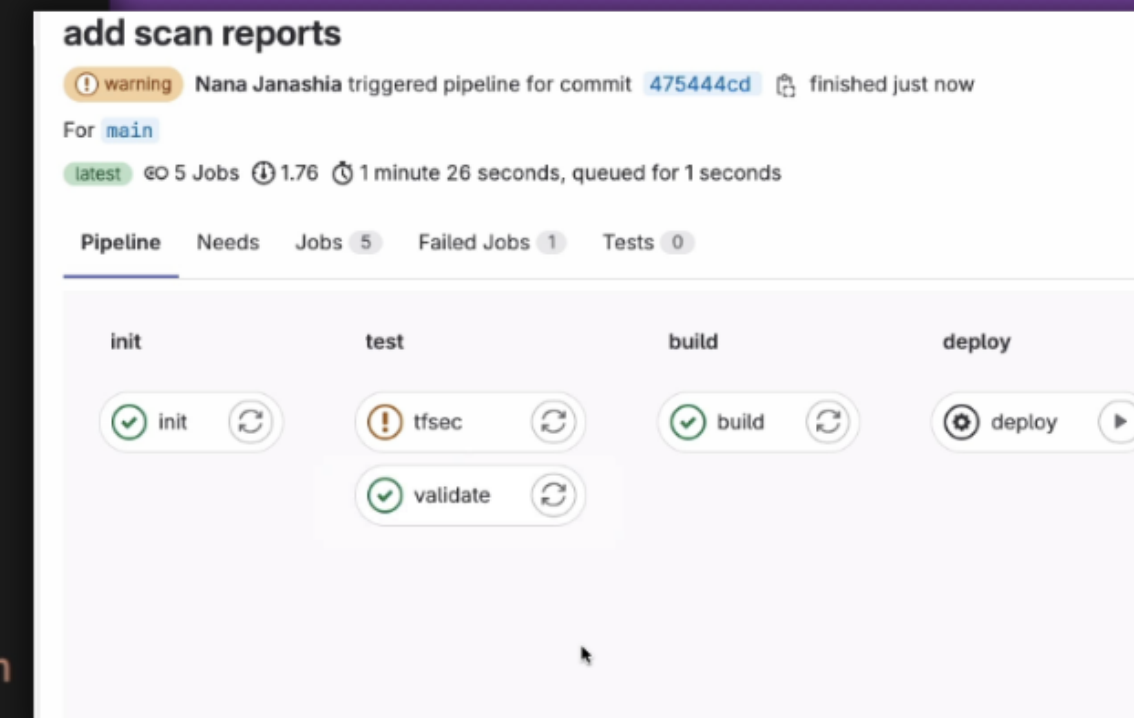
Using tfsec

- Open source security scanner for Terraform code
- Scans TF code for security vulnerabilities in infra configuration (static analysis)

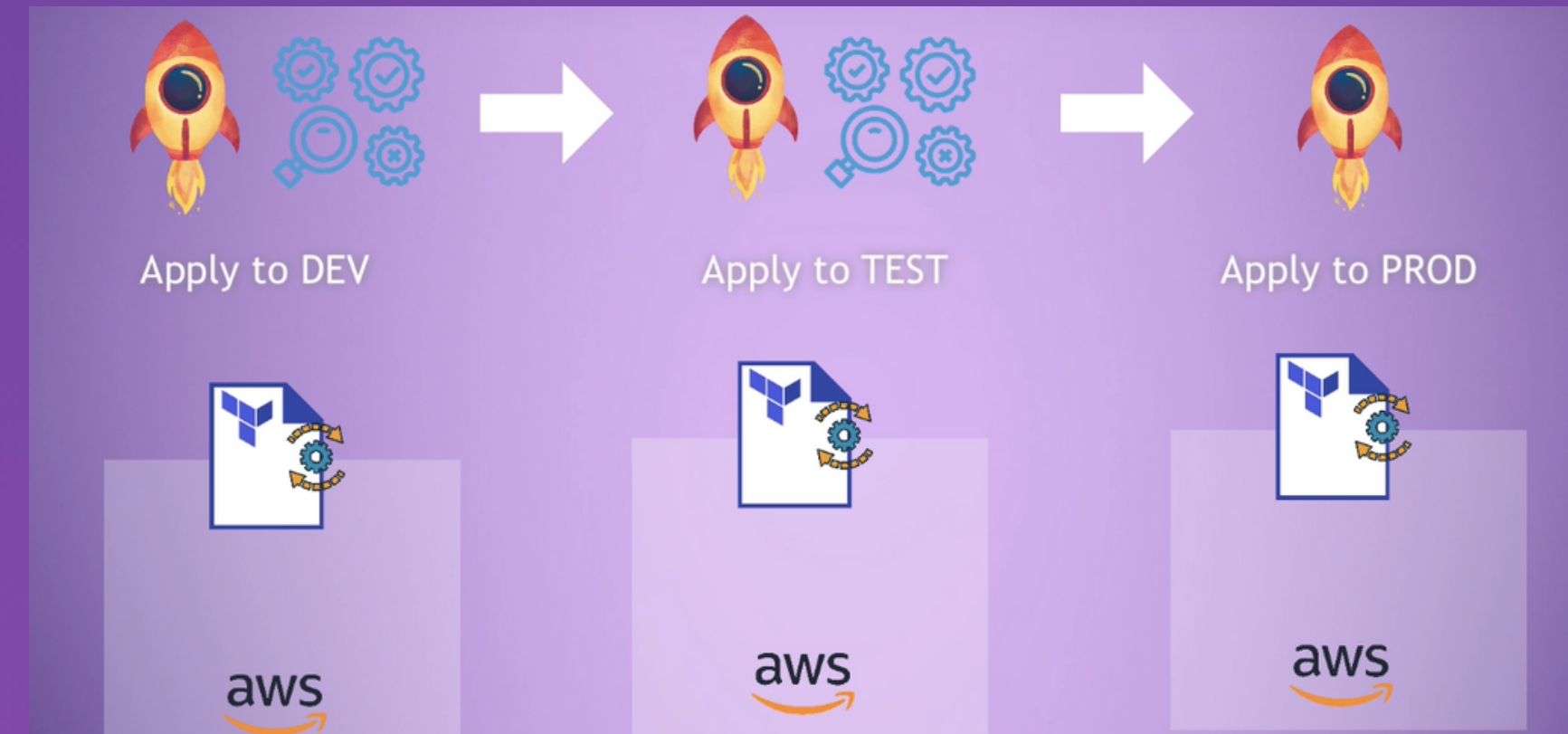
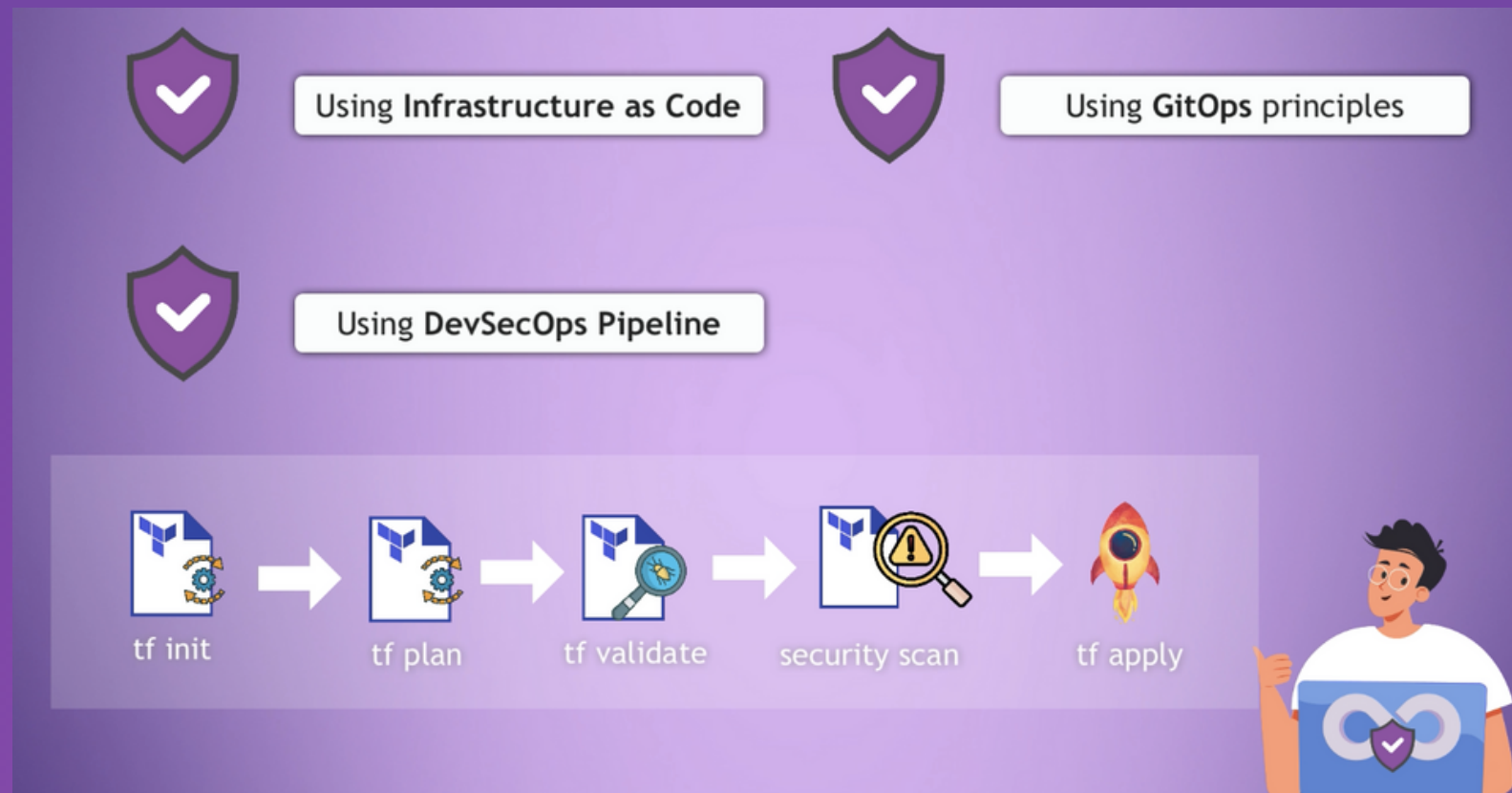


```
validate:
  stage: test
  script:
    - terraform validate
  allow_failure: true

tfsec:
  stage: test
  image:
    name: aquasec/tfsec:latest
    entrypoint: [ "" ]
  script:
    - tfsec . --format json --out tfsec.json
  allow_failure: true
  artifacts:
    when: always
    paths:
      - tfsec.json
```



Wrap Up





Cattle

vs



Pets



Treat servers as interchangeable resources that can be created, destroyed and replaced on demand



This approach is necessary, when we are working with IaC - recreating infrastructures from scratch with a clean state



Each server is carefully configured and monitored with individualized settings and configurations



Doesn't allow for dynamic, interchangeable environments