DevOps Infrastructure Documentation

Alex Brix

September 09, 2025

1 Level 1 – Web Console

- Login to AWS Web Console
- Launch EC2 instance using public AMI
- SSH access
- Manual configuration
- Manual deployment from Git

2 Level 2 – AWS CLI

- Use `aws ec2 run-instances`
- Transition from GUI to CLI
- Faster and repeatable flow

3 Level 3 – Infrastructure as Code

3.1 Iteration 1 – Basic Terraform

```
terraform init
terraform apply -auto-approve
```

3.2 Iteration 2 – Configuration with user-data

```
#!/bin/bash
sudo yum update -y
sudo yum install httpd -y
systemctl start httpd
systemctl enable httpd
```

3.3 Iteration 3 – Auto-deploy with user-data

Destroy everything *ightarrow* Add auto-deploy *ightarrow* Reconstruct *ightarrow* Verify

3.4 Iteration 4 – Ansible Integration

```
ansible-playbook -i inventory webserver.yml
```

3.5 Iteration 5 – Git-based Source

```
git clone https://github.com/alexbrix/infra.git
cd infra
terraform apply -auto-approve
```

3.6 Iteration 6 – Shared Backend

```
terraform {
  backend "s3" {
    bucket         = "alexbrix-terraform-state"
    key            = "infra/terraform.tfstate"
    region         = "eu-north-1"
    dynamodb_table = "terraform-lock-alexbrix"
  }
}
```

3.7 Iteration 7 – Optimization

```
terraform plan > plan.log
grep "No changes" plan.log && echo "Everything is fine"
```

3.8 Iteration 8 – Verification Script

```
#!/bin/bash
terraform plan > verify.log
cat verify.log | grep "No changes"
```

4 Level 4 – CI/CD and Security

4.1 Iteration 1 – CodePipeline + CloudFormation

```
aws cloudformation create-stack \
  --stack-name InfraStack \
  --template-body file://infra.yaml \
  --capabilities CAPABILITY_IAM
```

4.2 Iteration 2 – SSM Agent

```
sudo yum install -y amazon-ssm-agent
sudo systemctl start amazon-ssm-agent
aws ssm start-session --target i-xxxxxxxxxxxx
```

4.3 Iteration 3 – Security Hardening

```
aws ec2 revoke-security-group-ingress \
  --group-id sg-0a1b2c3d4e5f6g7h8 \
  --protocol tcp --port 22 --cidr 0.0.0.0/0
```

4.4 Iteration 4 – Automated Verification with CodeBuild

```
version: 0.2
phases:
  build:
    commands:
```

```
- echo "Verifying infrastructure..."
- terraform plan
```

5 Resources Used

- S3 Bucket: `alexbrix-terraform-state`
- DynamoDB Table: `terraform-lock-alexbrix`
- EC2 Instance:
  - Public IP: `13.53.124.87`
  - Private IP: `10.0.1.15`
  - Region: `eu-north-1`
  - Tag: `webserver-alexbrix`
  - Security Group: `sg-0a1b2c3d4e5f6g7h8`

6 Reflection

This iteration marks the transition from manual verification to automated workflows. I learned to use CodeBuild effectively, write and test buildspec.yml files, and apply secure flows using CloudFormation ChangeSets. This step improved both the security and efficiency of the infrastructure.