# DEVSECOPS

Nikola Krgović

Twin Star Systems

# ŠTA SVE OBUHVATA RAČUNARSKA BEZBEDNOST

- ČIME SE BAVI
- ZAŠTO SE TIME BAVI

# ZAŠTO JE BEZBEDNOST BITNA TEMA

- 
- 
-

# ZAŠTO JE BEZBEDNOST BITNA TEMA

- Štitimo svoje podatke
- Omogućavamo kontinuitet poslovanja (ISO 27001, ISO 23301)
- Čuvamo privatnost sebe i svojih korisnika
- Sprečavamo dalje širenje mallware-a

# DEFENCE IN DEPTH

- Zaštiti sistem na više nivoa
- Ako jedan sistem bude kompromitovan, tu je sledeći
- Cilj je sprečiti dalje širenje po sistemu – i po mreži

# TIPIČNE KOMPONENTE SISTEMA

- WEB SERVER : PREZENTACIONI SLOJ (NGINX)
- APLIKATIVNI SERVER (PHP, JAVA.... )
- BAZA PODATAKA (MYSQL , POSTGRESQL...)
- DODATNI SERVISI (REDIS, RABBITMQ, ELASTICSEARCH.....)

# MESTA ZAŠTITE

- 
- 
- 
- 
- 

# Mesta zaštite

- Regular updates
- Password protection : Use keys if you can
- Brute force protection : Fail2Ban
- System Hardening : sysctl options
- SSH hardening : users, passwords and keys
- File System : Mount options – noexec on /tmp and /var
- Network filtering : Firewall
- Database Hardening
- Mandatory Access Control : SE Linux – MAC
- DNS : Audit and remove unsafe queries

# Linux and Filesystem

- Mount /tmp as noexec
- Mount /var as noexec
- Change owner and group to PHP or NodeJS to match NginX
- Change owner / add write privileges to storage

```
# Network Security
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route =

# Disable ICMP redirect acceptance
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
```

# SE LINUX

- semanage fcontext -a -t httpd_sys_rw_content_t "/var/www/html/storage(/.*)?"
- restorecon -Rv /var/www/html/storage

- setsebool -P httpd_unified on
- setsebool -P httpd_can_network_memcache on
- setsebool -P httpd_can_network_connect_db on
- setsebool -P httpd_can_sendmail on

# SSH

Protocol 2

# Host keys - prefer modern algorithms

HostKey /etc/ssh/ssh_host_ed25519_key HostKey /etc/ssh/ssh_host_rsa_key HostKey /etc/ssh/ssh_host_ecdsa_key

# Key exchange algorithms (strongest first)

KexAlgorithms curve25519-sha256,curve25519-sha256@libssh.org,diffie-hellman-group-exchange-sha256

# Ciphers (AES-GCM preferred for hardware acceleration)

Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr

# MAC algorithms

MACs hmac-sha2-512-etm@openssh.com,hmac-sha2-256-etm@openssh.com,hmac-sha2-512,hmac-sha2-256

# Disable root login

PermitRootLogin no

PubkeyAuthentication yes

AuthorizedKeysFile .ssh/authorized_keys

PasswordAuthentication no

PermitEmptyPasswords no

# NGINX

```
# Security Headers
add_header X-Frame-Options "SAMEORIGIN" always;
add_header X-Content-Type-Options "nosniff" always;
add_header X-XSS-Protection "1; mode=block" always;
add_header Referrer-Policy "no-referrer-when-downgrade" always;
add_header Content-Security-Policy "default-src 'self' http: https: data: blob: 'unsafe-inline'" always;
 add_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload" always;
```

```
# Hide nginx version
server_tokens off;
more_clear_headers Server;

# Permissions Policy (formerly Feature Policy)
add_header Permissions-Policy "geolocation=(), microphone=(), camera=()" always;
```

# PHP

# Disable dangerous functions disable_functions = exec,passthru,shell_exec,system,proc_open,popen,curl_exec,curl_multi_exec,parse_ini_file,show_source,phpinfo,pcntl_exec,pcntl_fork,pcntl_signal,pcntl_waitpid,pcntl_wexitstatus,pcntl_wifexited,pcntl_wifsignaled,pcntl_wifstopped,pcntl_wstopsig,pcntl_wtermsig

# Session security

session.cookie_httponly = 1

session.cookie_secure = 1

session.cookie_samesite = "Strict"

session.use_strict_mode = 1

session.use_only_cookies = 1

session.use_trans_sid = 0

session.cookie_lifetime = 0

session.gc_maxlifetime = 1800

session.gc_probability = 1

session.gc_divisor = 100

session.sid_length = 48

session.sid_bits_per_character = 6

# REDIS

Command Override (redis.conf)

aclfile /etc/redis/users.acl r

rename-command FLUSHDB ""

rename-command FLUSHALL ""

rename-command KEYS ""

rename-command CONFIG "CONFIG_a8f5d2c9b1e3"

rename-command SHUTDOWN "SHUTDOWN_x7k2m9p4"

rename-command DEBUG ""

rename-command BGREWRITEAOF ""

rename-command BGSAVE ""

rename-command SAVE ""

rename-command SPOP ""

rename-command SREM ""

rename-command DEL "DEL_z3h8k5n2"

- *# Default user - disable it*
- user default off nopass nocommands

- *# Admin user with full access*
- user admin on >your-admin-password-here ~* +@all

- *# Application user with limited permissions*
- user app on >your-app-password-here ~* -@all +@read +@write +@fast -@dangerous +get +set +del +expire +ttl +exists +incr +decr +hget +hset +hdel +sadd +smembers +zadd +zrange

- *# Read-only user*
- user readonly on >your-readonly-password-here ~* -@all +@read +get +hget +hgetall +smembers +lrange +zrange +keys +scan

- *# Monitoring user*
- user monitor on >your-monitor-password-here ~* -@all +info +ping +slowlog +latency

- *# Replication user (if using replication)*
- user replica on >your-replica-password-here ~* -@all +psync +replconf +ping

# HOW DO WE KNOW

BEST WAY TO KNOW WHAT TO DO - USE TRIBAL KNOWLEDGE:

- HTTPS://DEV-SEC.IO/
- HTTPS://OWASP.ORG/WWW-PROJECT-TOP-TEN/

# AUTOMATION

Why Automate:

- 
- 
-

# Automation

Why Automate:

- Repeatability
- Mass deployements
- Code Review
- Self-Documenting

# DEVOPS AND DEVSECOPS

- What is DevOps?
- What is DevSecOps?
- Why is it not a job description?
- Whay should you care?

# AGILE

- What is Agile

# AGILE MANIFESTO

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

# DEVOPS

DevOps (Development + Operations) is a culture, set of practices, and tools that combine software development (Dev) and IT operations (Ops) to automate and integrate processes, enabling faster, more reliable software delivery. It breaks down silos between teams, emphasizing collaboration, continuous feedback, and automation (CI/CD) throughout the entire software lifecycle, from coding to deployment and monitoring, to deliver value to customers quicker and more consistently

# DEVOPS

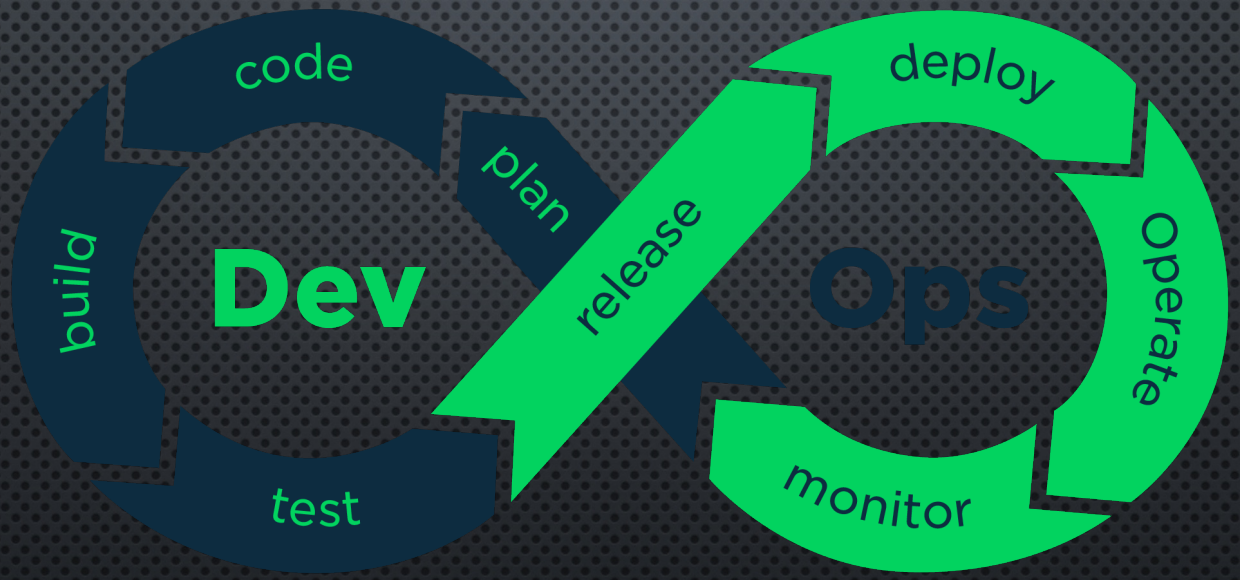
- Cultural Movement
- Differences from traditional methods
- Why is it different
- Why it matters
- How it affects the process
- How it affects you

# DEVOPS

Core Principles & Practices:

- **Culture**: Fosters shared responsibility, communication, and collaboration between Dev and Ops.
- **Automation**: Automates repetitive tasks in building, testing, and deploying code (CI/CD pipelines).
- **Continuous Integration/Delivery (CI/CD)**: Regularly merging code changes and automatically deploying them.
- **Monitoring & Feedback**: Continuously observing application performance and gathering feedback for rapid improvements.
- **Infrastructure as Code (IaC)**: Managing infrastructure through code, making it versionable and repeatable.

# DEVOPS




This Photo by Unknown Author is licensed under CC BY-SA

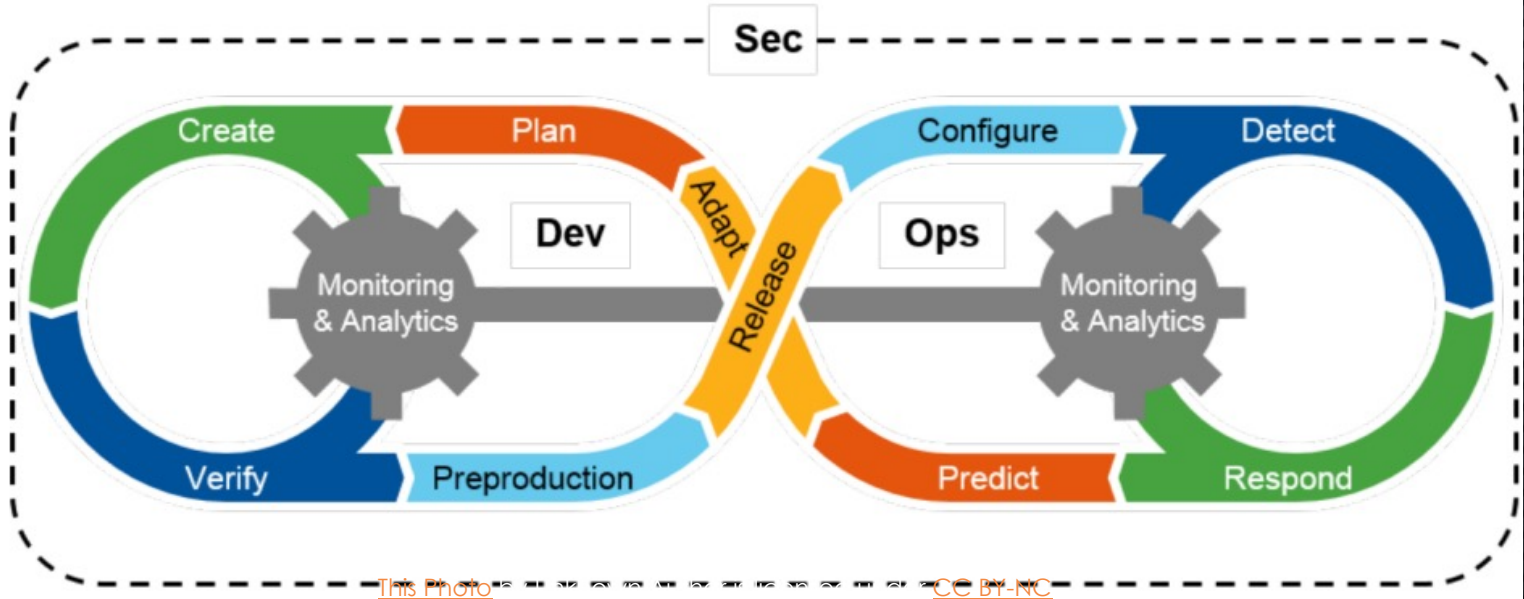
# DEVSECOPS

DevSecOps (Development, Security, & Operations) is a culture and practice that integrates security into every stage of the Software Development Lifecycle (SDLC), shifting security "left" to be a shared responsibility, not a final checkpoint. It uses automation and collaboration to embed security testing (SAST, DAST, SCA) directly into CI/CD pipelines, enabling teams to build and deliver secure, high-quality software faster and more efficiently, reducing costly late-stage vulnerabilities

# DevSecOps

- Secure by design
- Security on board from the start
- Security review of all code
- Secure code – and keep everything as code
- Security end-to-end

# DEVSECOPS




This Photo CC BY-NC

Source: Gartner (September 2016)

# DEVSECOPS

Key Principles

•**Shared Responsibility:** Security becomes everyone's job (devs, ops, security).

•**Automation:** Security checks are automated within pipelines, not done manually at the end.

•**Shift-Left Security:** Security is considered from the initial design/planning phase, not just before release.

•**Continuous Integration:** Security testing runs continuously with code builds

# DevSecOps

How It Works

- **Plan & Code:** Security requirements are defined; developers use secure coding practices.
- **Build:** Automated SAST (Static Application Security Testing) scans code for vulnerabilities.
- **Test:** DAST (Dynamic Application Security Testing) scans running applications; IAST (Interactive) provides deeper insights.
- **Deploy:** Automated security checks ensure only secure code moves forward.
- **Operate & Monitor:** Continuous monitoring for threats in production.

# TRADITIONAL AUTOMATION

BASH SCRIPTING

# Traditional Automation

Bash Scripting

- Fast to create
- Runs on any POSIX 1003 system
- Harder if it needs to take care of current state
- Lacks good ways to prevent configuration drift
- Best tool in some cases

# SCRIPT EXAMPLE

```
#!/bin/bash
Dnf update –y
Dnf install firewalld
Systemctl enable –now firewalld
firewall-cmd --enable-service=http –permanent
firewall-cmd --enable-service=https –permanent
Firewall-cmd --reload
```

# Modern approaches

## Agentic

- Runs on machine
- Uses resources (CPU, RAM)
- Keeps desired state
- Examples : Puppet, Chef

## Agentless

- Runs on centralised server
- No resources used
- Needs to be re-run
- Examples : Ansible, Salt

# Modern approaches

## Agentic

- ”Pets”

## Agentless

- “Cattle”

# Modern approaches

## Agentic

- "Pets"
- Databases
- Reliable storage

## Agentless

- "Cattle"
- Best for stateless
- Best if immutable

# PUPPET EXAMPLE

```
class rocky_setup {

  # Set the hostname
  exec { 'set_hostname':
    command => 'hostnamectl set-hostname myhost.mydomain.tld',
    unless  => "[ \"$(hostname)\" = \"myhost.mydomain.tld\" ]",
  }

  # Ensure required packages are installed
  package { [ 'firewalld', 'nginx', 'certbot' ]:
    ensure => installed,
  }

  # Start and enable firewalld
  service { 'firewalld':
    ensure => running,
    enable => true,
    require => Package['firewalld'],
  }

  # Configure FirewallD rules
  exec { 'allow_http':
    command => 'firewall-cmd --permanent --add-service=http',
    unless  => 'firewall-cmd --list-services | grep http',
    require => Service['firewalld'],
  }

  exec { 'allow_https':
    command => 'firewall-cmd --permanent --add-service=https',
    unless  => 'firewall-cmd --list-services | grep https',
    require => Service['firewalld'],
  }

  exec { 'allow_ssh_specific':
    command => 'firewall-cmd --permanent --add-rich-rule="rule family=ipv4 source address=1.2.3.4 service name=ssh accept"',
    unless  => 'firewall-cmd --list-rich-rules | grep 1.2.3.4',
    require => Service['firewalld'],
  }

  exec { 'reload_firewalld':
    command => 'firewall-cmd --reload',
    require => [ Exec['allow_http'], Exec['allow_https'], Exec['allow_ssh_specific'] ],
  }

  # Ensure Nginx service is running
  service { 'nginx':
    ensure => running,
    enable => true,
    require => Package['nginx'],
  }

  # Create Nginx vhost configuration
  file { '/etc/nginx/conf.d/myhost.mydomain.tld.conf':
```

# ANSIBLE EXAMPLE

```yaml
- name: Configure Rocky Linux 9 Server
  hosts: rocky_vm
  become: yes

  tasks:
    - name: Set the hostname
      ansible.builtin.hostname:
        name: "myhost.mydomain.tld"

    - name: Update /etc/hosts
      ansible.builtin.lineinfile:
        path: /etc/hosts
        line: "127.0.0.1 myhost.mydomain.tld"
        state: present

    - name: Install required packages
      ansible.builtin.dnf:
        name:
          - firewalld
          - nginx
          - certbot
        state: present

    - name: Start and enable firewalld
      ansible.builtin.systemd:
        name: firewalld
        enabled: yes
        state: started

    - name: Allow HTTP through firewall
      ansible.builtin.firewalld:
        service: http
        permanent: yes
        state: enabled
        immediate: yes

    - name: Allow HTTPS through firewall
      ansible.builtin.firewalld:
        service: https
        permanent: yes
        state: enabled
        immediate: yes

    - name: Allow SSH only from 1.2.3.4
      ansible.builtin.firewalld:
        rich_rule: 'rule family="ipv4" source address="1.2.3.4" service name="ssh" accept'
        permanent: yes
        state: enabled
        immediate: yes

    - name: Reload firewalld
      ansible.builtin.systemd:
        name: firewalld
        state: restarted

    - name: Create Nginx vhost config
      ansible.builtin.copy:
        dest: "/etc/nginx/conf.d/myhost.mydomain.tld.conf"
        content: |
```

# COMPARE PUPPET / ANSIBLE

Ansible Playbook - Ansible playbook files (ansible.json)

```yaml
---
- name: Configure Rocky Linux 9 Server
  hosts: rocky_vm
  become: yes

  tasks:
    - name: Set the hostname
      ansible.builtin.hostname:
        name: "myhost.mydomain.tld"

    - name: Update /etc/hosts
      ansible.builtin.lineinfile:
        path: /etc/hosts
        line: "127.0.0.1 myhost.mydomain.tld"
        state: present

    - name: Install required packages
      ansible.builtin.dnf:
        name:
          - firewalld
          - nginx
          - certbot
        state: present

    - name: Start and enable firewalld
      ansible.builtin.systemd:
        name: firewalld
        enabled: yes
        state: started

    - name: Allow HTTP through firewall
      ansible.builtin.firewalld:
        service: http
        permanent: yes
        state: enabled
        immediate: yes

    - name: Allow HTTPS through firewall
      ansible.builtin.firewalld:
        service: https
        permanent: yes
        state: enabled
        immediate: yes

    - name: Allow SSH only from 1.2.3.4
      ansible.builtin.firewalld:
        rich_rule: 'rule family="ipv4" source address="1.2.3.4" service name="ssh" accept'
        permanent: yes
        state: enabled
        immediate: yes

    - name: Reload firewalld
      ansible.builtin.systemd:
        name: firewalld
        state: restarted

    - name: Create Nginx vhost config
      ansible.builtin.copy:
        dest: "/etc/nginx/conf.d/myhost.mydomain.tld.conf"
        content: |
          server {
              listen 80;
              server_name myhost.mydomain.tld;
              root /var/www/myhost;
              index index.html;
```

```ruby
class rocky_setup {

  # Set the hostname
  exec { 'set_hostname':
    command => 'hostnamectl set-hostname myhost.mydomain.tld',
    unless  => "[ \"${hostname)\" = \"myhost.mydomain.tld\" ]",
  }

  # Ensure required packages are installed
  package { [ 'firewalld', 'nginx', 'certbot' ]:
    ensure => installed,
  }

  # Start and enable firewalld
  service { 'firewalld':
    ensure => running,
    enable => true,
    require => Package['firewalld'],
  }

  # Configure FirewallD rules
  exec { 'allow_http':
    command => 'firewall-cmd --permanent --add-service=http',
    unless  => 'firewall-cmd --list-services | grep http',
    require => Service['firewalld'],
  }

  exec { 'allow_https':
    command => 'firewall-cmd --permanent --add-service=https',
    unless  => 'firewall-cmd --list-services | grep https',
    require => Service['firewalld'],
  }

  exec { 'allow_ssh_specific':
    command => 'firewall-cmd --permanent --add-rich-rule="rule family=ipv4 source address=1.2.3.4 service name=ssh accept"',
    unless  => 'firewall-cmd --list-rich-rules | grep 1.2.3.4',
    require => Service['firewalld'],
  }

  exec { 'reload_firewalld':
    command => 'firewall-cmd --reload',
    require => [ Exec['allow_http'], Exec['allow_https'], Exec['allow_ssh_specific'] ],
  }

  # Ensure Nginx service is running
  service { 'nginx':
    ensure => running,
    enable => true,
    require => Package['nginx'],
  }

  # Create Nginx vhost configuration
  file { '/etc/nginx/conf.d/myhost.mydomain.tld.conf':
    ensure  => file,
    content => """
      server {
          listen 80;
          server_name myhost.mydomain.tld;
          root /var/www/myhost;
          index index.html;

          location / {
              try_files $uri $uri/ =404;
          }
      }
    """,
    require => Package['nginx'],
```

# Automation for the cloud

## Cloud Native

- Cloud Formation
- ARM templates
- BICEP

## Cloud agnostic

- Terraform
- Open Tofu
- Pulumi

# WHY TERRAFORM / OPEN TOFU

- Modules for each cloud
- Supports AWS, Google Cloud, Azure, Oracle Cloud, Huawei Cloud…
- Stores state in common storage – like S3
- Can run in a git pipeline

# TERRAFORM EXAMPLE : AWS

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_vpc" "main" {
  cidr_block = "10.10.0.0/16"
  enable_dns_support = true
  enable_dns_hostnames = true
}

resource "aws_subnet" "main" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = "10.10.1.0/24"
  availability_zone = "us-east-1c"
  map_public_ip_on_launch = true
}

resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.main.id
}

resource "aws_route_table" "public" {
  vpc_id = aws_vpc.main.id
}

resource "aws_route" "internet_access" {
  route_table_id         = aws_route_table.public.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.gw.id
}

resource "aws_route_table_association" "public" {
  subnet_id      = aws_subnet.main.id
  route_table_id = aws_route_table.public.id
}

resource "aws_eip" "nat" {
  domain = "vpc"
}

resource "aws_nat_gateway" "nat" {
  allocation_id = aws_eip.nat.id
  subnet_id     = aws_subnet.main.id
}

resource "aws_security_group" "sg" {
  vpc_id = aws_vpc.main.id

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["1.2.3.4/32"]
```

# TERRAFORM EXAMPLE : AWS AND AZURE

```
provider "aws" {
  region = "us-east-1"
}

resource "aws_vpc" "main" {
  cidr_block = "10.10.0.0/16"
  enable_dns_support = true
  enable_dns_hostnames = true
}

resource "aws_subnet" "main" {
  vpc_id            = aws_vpc.main.id
  cidr_block        = "10.10.1.0/24"
  availability_zone = "us-east-1c"
  map_public_ip_on_launch = true
}

resource "aws_internet_gateway" "gw" {
  vpc_id = aws_vpc.main.id
}

resource "aws_route_table" "public" {
  vpc_id = aws_vpc.main.id
}

resource "aws_route" "internet_access" {
  route_table_id         = aws_route_table.public.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.gw.id
}

resource "aws_route_table_association" "public" {
  subnet_id      = aws_subnet.main.id
  route_table_id = aws_route_table.public.id
}

resource "aws_eip" "nat" {
  domain = "vpc"
}

resource "aws_nat_gateway" "nat" {
  allocation_id = aws_eip.nat.id
  subnet_id     = aws_subnet.main.id
}

resource "aws_security_group" "sg" {
  vpc_id = aws_vpc.main.id

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["1.2.3.4/32"]
```

```
provider "azurerm" {
  features {}
}

resource "azurerm_resource_group" "main" {
  name     = "my-resource-group"
  location = "eastus"
}

resource "azurerm_virtual_network" "main" {
  name                = "my-vnet"
  address_space       = ["10.10.0.0/16"]
  location            = azurerm_resource_group.main.location
  resource_group_name = azurerm_resource_group.main.name
}

resource "azurerm_subnet" "main" {
  name                 = "my-subnet"
  resource_group_name  = azurerm_resource_group.main.name
  virtual_network_name = azurerm_virtual_network.main.name
  address_prefixes     = ["10.10.1.0/24"]
}

resource "azurerm_public_ip" "nat" {
  name                = "nat-ip"
  location            = azurerm_resource_group.main.location
  resource_group_name = azurerm_resource_group.main.name
  allocation_method   = "Static"
}

resource "azurerm_nat_gateway" "nat" {
  name                = "my-nat-gateway"
  location            = azurerm_resource_group.main.location
  resource_group_name = azurerm_resource_group.main.name
}

resource "azurerm_nat_gateway_public_ip_association" "nat" {
  nat_gateway_id       = azurerm_nat_gateway.nat.id
  public_ip_address_id = azurerm_public_ip.nat.id
}

resource "azurerm_network_security_group" "sg" {
  name                = "my-security-group"
  location            = azurerm_resource_group.main.location
  resource_group_name = azurerm_resource_group.main.name
}

resource "azurerm_network_security_rule" "http" {
  name                        = "allow-http"
  priority                    = 1001
  direction                   = "Inbound"
  access                      = "Allow"
  protocol                    = "Tcp"
  source_port_range           = "*"
  destination_port_range      = "80"
  source_address_prefix       = "*"
  destination_address_prefix  = "*"
  network_security_group_name = azurerm_network_security_group.sg.name
  resource_group_name         = azurerm_resource_group.main.name
}

resource "azurerm_network_security_rule" "https" {
  name                        = "allow-https"
  priority                    = 1002
  direction                   = "Inbound"
  access                      = "Allow"
  protocol                    = "Tcp"
```

# CODE REVIEW

```
nkrgovic@Nikolas-MBP-4 viv % tfsec ./scan

  aws.tf:53

   46   resource "aws_security_group" "sg" {
   ..
   53 [     cidr_blocks = ["0.0.0.0/0"]
   ..
   76   }

          ID aws-ec2-no-public-ingress-sgr
      Impact Your port exposed to the internet
  Resolution Set a more restrictive cidr range

  More Information
  - https://aquasecurity.github.io/tfsec/v1.28.14/checks/aws/ec2/no-public-ingress-sgr/
  - https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/security_group_rule#cidr_blocks


Result #2 CRITICAL Security group rule allows ingress from public internet.

  aws.tf:60

   46   resource "aws_security_group" "sg" {
   ..
   60 [     cidr_blocks = ["0.0.0.0/0"]
   ..
   76   }

          ID aws-ec2-no-public-ingress-sgr
      Impact Your port exposed to the internet
  Resolution Set a more restrictive cidr range

  More Information
  - https://aquasecurity.github.io/tfsec/v1.28.14/checks/aws/ec2/no-public-ingress-sgr/
  - https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/security_group_rule#cidr_blocks


Result #3 CRITICAL Security group rule allows egress to multiple public internet addresses.

  aws.tf:74

   46   resource "aws_security_group" "sg" {
   ..
   74 [     cidr_blocks = ["0.0.0.0/0"]
   ..
   76   }

          ID aws-ec2-no-public-egress-sgr
      Impact Your port is egressing data to the internet
  Resolution Set a more restrictive cidr range

  More Information
  - https://aquasecurity.github.io/tfsec/v1.28.14/checks/aws/ec2/no-public-egress-sgr/
  - https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/security_group
```

# Tools to use

- Do not reinvent the wheel
- Rely on tribal knowledge
- Use available resources – like ansible galaxy
- Focus on well know authors :
  - Go to vendor (use nginx module to setup nginx server)
  - Known authors like GeerlingGuy (Jeff Geerling)

# Workshop Topic

- Create a simple ansible
- Focus on hardening a Linux setup
- Use Rocky Linux, try to keep SE Linux enforcing
- Add as many enhancements as you can

# CODE EXAMPLE : HTTPS://GITHUB.COM/NKRGOVIC/DEVSECOPS

setup-nginx.yml > ...

```yaml
---
- name: Setup Nginx on Rocky Linux 9 with Security Hardening
  hosts: rocky_servers
  become: yes

  tasks:
    - name: Install required packages
      ansible.builtin.dnf:
        name:
          - nginx
          - firewalld
          - python3-libselinux
          - python3-policycoreutils
        state: present

    - name: Ensure SELinux is enabled and enforcing
      ansible.posix.selinux:
        policy: targeted
        state: enforcing
      register: selinux_result

    - name: Check if reboot is required for SELinux
      ansible.builtin.debug:
        msg: "SELinux mode changed. A reboot may be required."
      when: selinux_result.reboot_required | default(false)

    - name: Add noexec to /var mount point in fstab
      ansible.posix.mount:
        path: /var
        src: "{{ ansible_mounts | selectattr('mount', 'equalto', '/var') | map(attribute='device') | first | default(omit) }}"
        fstype: "{{ ansible_mounts | selectattr('mount', 'equalto', '/var') | map(attribute='fstype') | first | default('xfs') }}"
        opts: "{{ ansible_mounts | selectattr('mount', 'equalto', '/var') | map(attribute='options') | first | default('defaults') }},noexec"
        state: mounted
      when: ansible_mounts | selectattr('mount', 'equalto', '/var') | list | length > 0

    - name: Add noexec to /tmp mount point in fstab
      ansible.posix.mount:
        path: /tmp
        src: "{{ ansible_mounts | selectattr('mount', 'equalto', '/tmp') | map(attribute='device') | first | default('tmpfs') }}"
        fstype: "{{ ansible_mounts | selectattr('mount', 'equalto', '/tmp') | map(attribute='fstype') | first | default('tmpfs') }}"
        opts: "{{ ansible_mounts | selectattr('mount', 'equalto', '/tmp') | map(attribute='options') | first | default('defaults') }},noexec"
        state: mounted
      when: ansible_mounts | selectattr('mount', 'equalto', '/tmp') | list | length > 0

    - name: Create /tmp with noexec if not a separate mount
      ansible.posix.mount:
        path: /tmp
        src: tmpfs
        fstype: tmpfs
        opts: defaults,noexec,nosuid,nodev
        state: mounted
      when: ansible_mounts | selectattr('mount', 'equalto', '/tmp') | list | length == 0

    - name: Start and enable firewalld
      ansible.builtin.systemd:
        name: firewalld
        state: started
        enabled: yes

    - name: Ensure firewalld uses nftables backend
      ansible.builtin.lineinfile:
        path: /etc/firewalld/firewalld.conf
        regexp: '^FirewallBackend='
        line: 'FirewallBackend=nftables'
        state: present
      notify: Restart firewalld
```

# Workshop Topic

- Create a simple ansible
- Focus on hardening a Linux setup
- Use Rocky Linux, try to keep SE Linux enforcing
- Add as many enhancements as you can
  - Add SysCTL enhancements
  - Install Nginx, PHP-FPM, MySQL
  - Harden NginX
  - Harden MySQL
  - Add FirewallD
  - Harden SSH
  - Add Web Application Firewall
  - Add more – as you see fit….

# EXAMPLE OF SOLUTION

- CODE EXAMPLE : HTTPS://GITHUB.COM/NKRGOVIC/DEVSECOPS

# QUESTIONS ?

# HVALA!