Git

```
# Git Documentation
# Git global setup
git config --global user.name "John Doe"
git config --global user.email
"johndoe@gmail.com"

# Clone and Edit a repository
git clone
git@git.example.com:repository/project.git
cd project
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master

# Convert existing folder to repo and push
cd existing_folder
git init
git remote add origin
git@gitexample.com:repository/project.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

Git Secrets

```
# Git-Secrets prevents you from committing
secrets/credentials into git repositories
# Scan for secrets on each commit
git secrets —install /path/to/files
git secrets —register-aws

# Scan file/folders for secrets
git secrets --scan /path/to/file
git secrets --scan -r /path/to/directory

# Adds a prohibited pattern to the current
repo:
git secrets --add '[A-Z0-9]{20}'

# Adds a prohibited pattern to the global git
config:
git secrets --add --global '[A-Z0-9]{20}'

# Add an allowed pattern:
git secrets --add -a 'allowed pattern'
```

Scout Suite

```
# Scout Suite is a multi-cloud audit tool
# Install and configure
git clone https://github.com/nccgroup/ScoutSuite
cd ScoutSuite
sudo pip3 install -r requirements.txt
python3 scout.py --help #Check install

# Pull the latest ruleset
curl
https://raw.githubusercontent.com/nccgroup/Scout
Suite/master/ScoutSuite/providers/aws/rules/rule
sets/detailed.json > detailed-rules.json

# Run with the latest Ruleset
python3 scout.py aws --profile <profile> --
ruleset <ruleset>
```

Docker

```
# Docker Documentation
docker pull <image>:<tag>
docker build -f /path/dockerfile -t imagename .

docker image ls
docker image rm <imageid>
docker commit containerid [REPOSITORY[:TAG]]
docker container ls -a
docker container prune
docker info
docker kill <containerid>
docker rm <containerid>

# Bulk Delete All Containers
docker ps -a -q | xargs -n 1 -I {} docker rm {}

# Run Containers in Detached Mode
docker run -d -p 80:80 myimage nginx -g 'daemon
off;'

# Run Interactive Containers with Mounted Files
docker run -v /hostpath:/containerpath -it
<image>:<tag>

docker save image:tag > image.tar
```

SANS CLOUD SECURITY

Cloud Security and DevOps "Fix Security Issues Left of Prod"

By Ross Young

Cheat Sheet v1.1.4

SANS.ORG/CLOUD-SECURITY

Docker Security Checks

Dockle - Check your Dockerfile against the CIS Benchmarks with a Container Image Linter

```
# Install Dockle
https://github.com/goodwithtech/dockle

dockle REPOSITORY/IMAGE:TAG
dockle --exit-code 1 —exit-level fatal
IMAGE:TAG
```

Docker Scan - Find Vulnerabilities within a Container Image

```
docker scan --file /Path/Dockerfile IMAGE:TAG
```

Container Vuln Scan (Excluding the Base Image)

```
docker scan --file /Path/Dockerfile --exclude-
base IMAGE:TAG
```

Dependency Tree

```
docker scan --dependency-tree IMAGE:TAG
```

Docker-Bench - Evaluate your Docker Engine configuration against the CIS Benchmark
Install Go, then clone this repository
https://github.com/aquasecurity/docker-bench

```
go build -o docker-bench .

./docker-bench
```

Terraform Syntax

Blocks are the configuration of an object

Arguments assign a value to a name.

Expressions represent a value, either literally or by referencing and combining other values.

```
<BLOCK TYPE> "<BLOCK LABEL>" "<BLOCK LABEL>" {
  # Block body
  <IDENTIFIER> = <EXPRESSION> # Argument
}

# Example:

resource "aws_vpc" "main" {
  cidr_block = var.base_cidr_block
}
```

CloudFormation (YAML Syntax)

```
Resources:
  Logical ID:
    Type: Resource type
    Properties:
      Set of properties

# Example

Resources:
  MyInstance:
    Type: "AWS::EC2::Instance"
    Properties:
      UserData:
        "Fn::Base64":
          !Sub |
            Queue=${MyQueue}
      AvailabilityZone: "us-east-1a"
      ImageId: "ami-0ff8a91507f77f867"
  MyQueue:
    Type: "AWS::SQS::Queue"
    Properties: {}
```

Infrastructure Scans (Terraform, CloudFormation, & Helm)

Terrascan is a misconfiguration scanner. It can scan Terraform, Kubernetes, and other file types.

```
git clone git@github.com:accurics/terrascan.git
cd terrascan
make build
./bin/terrascan
terrascan scan -t aws

# Find security misconfigurations in Helm Charts
terrascan scan -I helm
```

Checkov looks for misconfigurations in files such as Terraform, Cloud Formation, and even Helm Charts.

```
pip install checkov
checkov -f /path/example.tf

# Find security misconfigurations in Helm Charts
checkov --framework kubernetes -d <template files>
```

CFN NAG looks for misconfigurations in CloudFormation templates.

```
gem install cfn-nag
cfn_nag_scan --input-path <path to templates>
```

Azure Key Store

```
# Create a Resource Group
az group create --name “MyResourceGroup” -l “EastUS”

# Create a new key in the keyvault
az keyvault create --name “<unique name>” --resource-group “MyResourceGroup” --location “EastUS”

# Show details of a key vault
az keyvault show --name MyKeyVault

# List Azure Key Vaults
az keyvault list --resource-group “MyResourceGroup”

# Delete a Key Vault
az keyvault delete --name MyKeyVault --resource-group MyResourceGroup
```

AWS Systems Manager Parameter Store

```
aws ssm put-parameter --name MyParameter --value "secret_value" --type SecureString

aws ssm get-parameter --name MyParameter --with-decryption
```

Jenkins Integration

Scan is a free open-source audit tool for DevOps teams. It can perform:

- Credentials Scanning to detect accidental secret leaks
- Static Analysis Security Testing (SAST) for a range of languages and frameworks
- Open-source dependencies audit
- License violation checks

You can add the following stage to your Jenkinsfile (declarative syntax) for basic integrations

```
stages {
    stage('Scan') {
        agent {
            docker { image 'shiftleft/sast-scan' }
        }
        steps {
            sh 'scan'
        }
    }
}
```