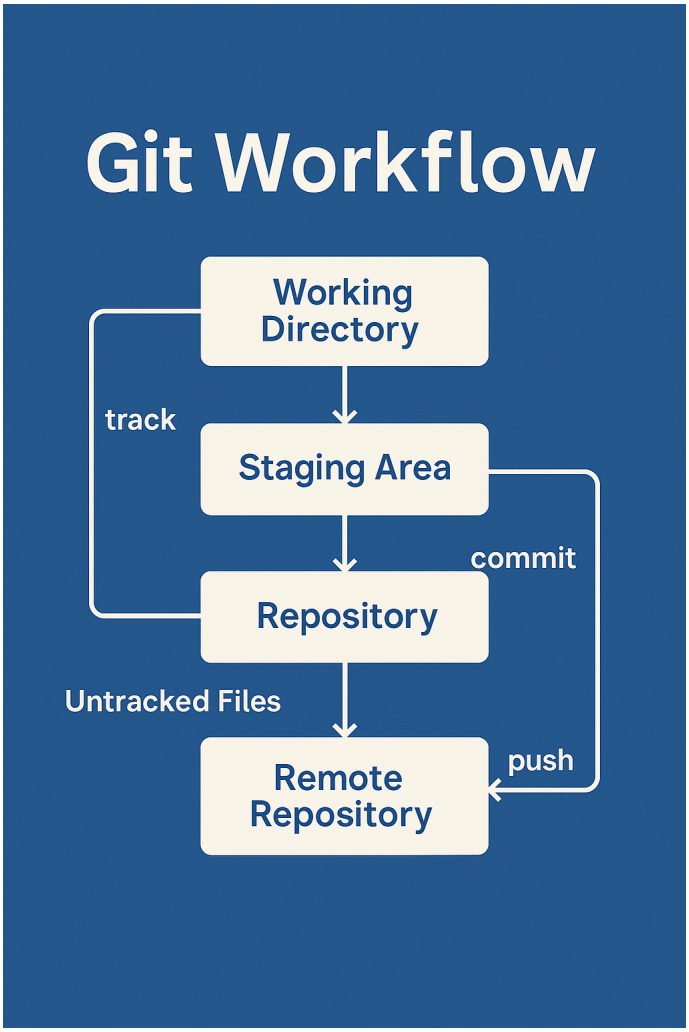
GIT FOR NETWORK ENGINEER

GIT BASICS

4-MAY-2025

STRUCTURE

GIT



INTRODUCTION - WHAT IS GIT?

In today's networked world, version control isn't just for developers. With infrastructure-as-code, automation, and programmable networks, Git has become essential for network engineers. This guide explores how Git can transform your workflow, prevent misconfigurations, and enhance collaboration.

WHY NETWORK ENGINEERS NEED GIT

- Track configuration changes across devices.
- Collaborate on network automation scripts.
- Maintain history of infrastructure modifications.
- Use Git to roll back broken changes quickly.

GIT WORKFLOW FOR NETWORK ENGINEERS

1. Pull latest changes.
2. Make edits (script/config).
3. Test in lab.
4. Commit and push to central repo.
5. Create Pull Request for review (optional).

WHAT IS GIT?

Git is a distributed version control system that allows multiple users to manage code or text-based files collaboratively. It tracks changes, supports branches, and stores full project history locally and remotely.

GIT VS GITHUB VS GITLAB

- **Git:** Command-line tool for version control.
- **GitHub/GitLab:** Web-based platforms for hosting Git repositories, collaboration, and CI/CD integration.
- **Self-hosted options:** GitLab CE or Gitea for enterprise privacy.

BASIC GIT TERMINOLOGY

Repository: Directory tracked by Git.

Commit: Snapshot of changes.

Branch: Parallel line of development.

Clone: Copy of a remote repository.

Merge: Combine branches.

Push/Pull: Upload/download changes.

INSTALLING GIT

Windows/macOS/Linux:

Download: https://git-scm.com/

Or install via terminal:

```
sudo apt install git      # Debian/Ubuntu
sudo yum install git      # RedHat/CentOS
brew install git          # macOS
```

INITIAL GIT CONFIGURATION

Set your identity:

```
git config --global user.name "Your Name"
git config --global user.email your@email.com
```

Check your config:

```
git config –list
```

CREATING YOUR FIRST REPO

```
mkdir net-automation

cd net-automation

git init

touch readme.md

git add readme.md

git commit -m "Initial commit"
```

CLONING AN EXISTING REPO

```
git clone https://github.com/your-org/network-scripts.git

cd network-scripts
```

Use this to collaborate or download shared configuration files.

TRACKING CONFIGURATION CHANGES

Use Git to track differences in router config templates:

```
git diff
```

This helps identify unintended changes quickly.

USING BRANCHES FOR LAB VS PRODUCTION

Keep lab testing isolated:

```
git checkout -b lab-testing
```

Once tested:

```
git checkout main

git merge lab-testing
```

MERGING AND RESOLVING CONFLICTS

When Git detects changes to the same lines, it prompts you to manually resolve:

```
git status

git diff
```

Edit files, then:

```
git add .

git commit
```

STASHING UNCOMMITTED WORK

Quickly switch tasks without losing changes:

```
git stash

# Later

git stash pop
```

Ideal for production outages or priority tasks.

TAGGING STABLE CONFIGURATIONS

```
git tag -a v1.0 -m \"First tested deployment\"

git push origin v1.0
```

Use tags to mark stable releases of automation scripts or config snapshots.

ROLLING BACK CHANGES

To revert to a specific state:

```
git log

git checkout <commit_hash>
```

Use with caution or on branches.

AUTOMATING BACKUPS WITH GIT

Automate daily config backups:

```
# Script:

netmiko_pull.py > backup.conf

git add backup.conf

git commit -m \"Daily backup\"

git push
```

INTEGRATING GIT WITH ANSIBLE & NETMIKO

- Store Ansible playbooks in Git.
- Store Netmiko Python scripts with version tracking.
- Share credentials and inventory templates securely.

GIT + CI/CD FOR NETWORK AUTOMATION

Trigger jobs on git push:

- Run tests (syntax, ping, path trace).
- Deploy config changes.
- Alert via Slack/Email.

Use tools: GitLab CI, Jenkins, CircleCI.

COLLABORATING AS A TEAM

- Use **Pull Requests** to review code/configs.
- Assign reviewers and approve changes.
- Track issues or feature requests using GitHub Issues.

SECURE YOUR REPOSITORIES

Use `.gitignore` to exclude secrets:

```
*.log

*.pem

passwords.yaml
```

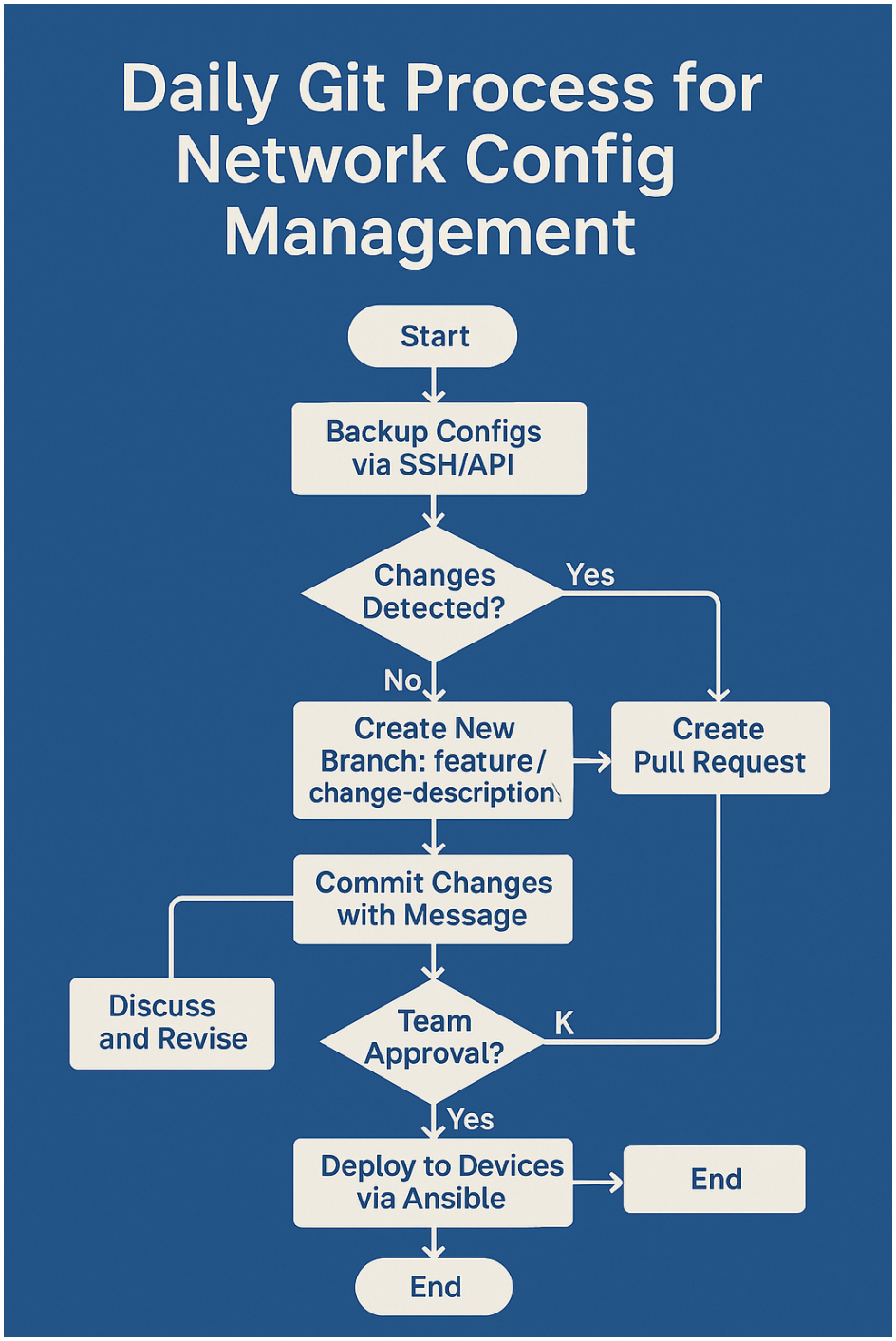
Use GitHub Secrets for automation tokens.

BOOST EFFICIENCY WITH GIT ALIASES

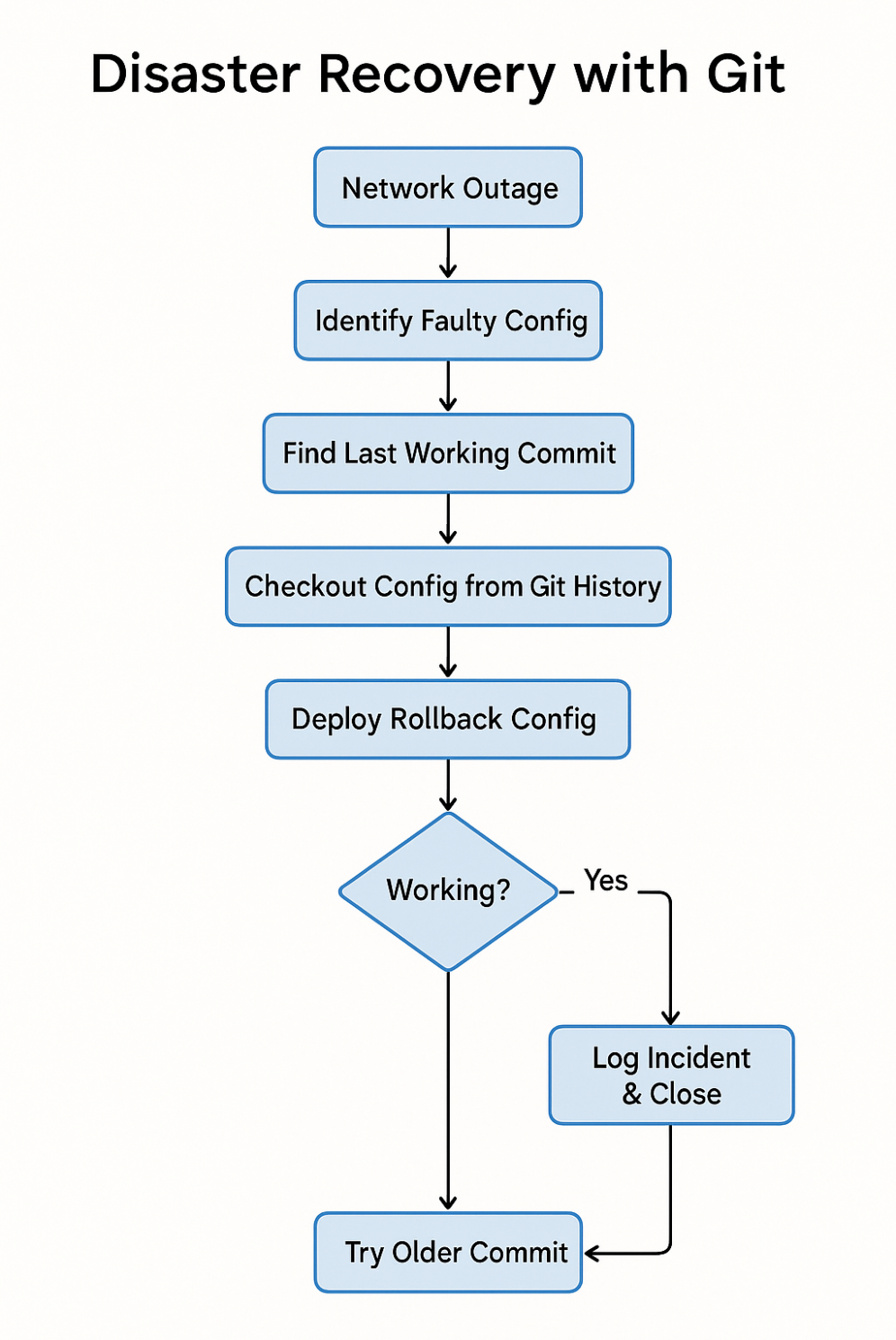
```
git config --global alias.st status

git config --global alias.co checkout

git config --global alias.br branch
```

GIT WORKFLOW DIAGRAM

a. Daily GIT process for network config management



b. Disaster recovery with GIT



EXAMPLE: NETMIKO SCRIPT - VLAN_CONFIG_NETMIKO.PY

```
from netmiko import ConnectHandler

# Device details

device = {

    "device_type": "cisco_ios",

    "host": "192.168.1.10",

    "username": "admin",

    "password": "cisco123",

}

# VLAN configuration

vlan_id = "50"

vlan_name = "Mgmt_VLAN"

commands = [

    f"vlan {vlan_id}",

    f"name {vlan_name}",

    "exit"

]

# Connect and send config

try:

    net_connect = ConnectHandler(**device)

    output = net_connect.send_config_set(commands)

    print(output)

    net_connect.disconnect()

except Exception as e:

    print(f"Error: {e}")
```

GIT REPOSITORY STRUCTURE EXAMPLE

```
network-configs-git/


├── README.md
├── .gitignore
├── scripts/
│   └── vlan_config_netmiko.py
├── configs/
│   └── switch01_backup_2024-05-04.cfg
├── docs/
│   └── vlan_strategy.md
└── logs/
    └── vlan_config_log.txt
```

GIT WORKFLOW EXAMPLE

1. **Clone your repo (first time):**

git clone https://github.com/yourname/network-configs-git.git

cd network-configs-git

2. **Create a branch for the new VLAN config:**

git checkout -b feature/add-vlan-50

3. **Add your script:**

cp ~/Downloads/vlan_config_netmiko.py scripts/

git add scripts/vlan_config_netmiko.py

4. **Commit with a message:**

git commit -m "Add Netmiko script for VLAN 50 creation"

5. **Push to GitHub:**

git push origin feature/add-vlan-50

6. **Open a Pull Request for review.**

REAL-WORLD USE CASE

A network team used Git to manage over 500+ device configs, implementing version tracking and rollback. They reduced configuration errors by 70% and accelerated change audits during outages.

CONCLUSION: EMBRACING GIT FOR MODERN NETWORK ENGINEERING

As networks become increasingly programmable and distributed, traditional ways of managing device configurations no longer scale. Git provides network engineers with a powerful, structured, and collaborative approach to manage configurations, scripts, and infrastructure-as-code. It eliminates guesswork, enables rollback, and brings transparency to every change made across your environment.

By incorporating Git into your daily workflow, you gain:

- **Version Control:** Track changes, revert to stable states, and audit history with precision.
- **Collaboration:** Work with your team on automation scripts, playbooks, and policies seamlessly.
- **Automation Integration:** Combine Git with tools like Netmiko, Ansible, and CI/CD for consistent deployments.
- **Disaster Recovery:** Swiftly recover from outages by restoring known-good configurations from Git history.

No matter your current skill level, starting small—by versioning backup configs or storing Netmiko scripts—is a meaningful first step. Over time, Git will become the backbone of your NetDevOps journey.