

CI/CD Pipelines for Infrastructure Automation

cisco Live !

Adrian ILIESIU
Developer Advocate, @aidevnet

Session ID: BRKDEV-2117

Cisco Webex App

Questions?

Use Cisco Webex App to chat with the speaker after the session

How

- 1 Find this session in the Cisco Live Mobile App
- 2 Click “Join the Discussion”
- 3 Install the Webex App or go directly to the Webex space
- 4 Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until 14 November 2025.



<https://ciscolive.ciscoevents.com/ciscolivebot/#BRKDEV-2117>

Agenda

- 01 **CI/CD Fundamentals: Understanding Continuous Integration & Delivery**
- 02 **Infrastructure Automation: Applying CI/CD to Networks**
- 03 **GitLab Deep Dive: Your DevOps Platform for Infrastructure**
- 04 **Live Demonstration: Seeing CI/CD in Action**
- 05 **Conclusion: Challenges & Best Practices**
- 06 **Resources: Further Learning & Engagement**

CI/CD Fundamentals

CI/CD Fundamentals: Continuous Integration (CI)

- **Definition:** The practice of frequently merging code changes into a central repository.
- **Process:**
 - Developers work on separate branches for features/fixes.
 - Code is regularly merged into a main branch.
 - Automated tests run immediately upon merge.
- **Benefits:**
 - Early detection of integration issues.
 - Improved code quality and stability.
 - Faster feedback loops for developers.

CI/CD Fundamentals: Continuous Delivery (CD)

- **Definition:** Ensuring software is always in a deployable state after CI.
- **Process:**
 - Automated builds and tests confirm integration requirements.
 - Artifacts are pushed to a "staging" or "pre-production" environment.
 - Additional automated tests (e.g., performance, security) are executed.
- **Benefits:**
 - Reliable and repeatable release process.
 - Confidence that changes are ready for production at any time.
 - Reduced risk during deployments.

CI/CD Fundamentals: Continuous Deployment (CD)

- **Definition:** Automatically deploying all changes that pass CI/CD to production.
- **Process:**
 - Extends Continuous Delivery by automating the final push to production.
 - Leverages various deployment strategies (e.g., Blue/Green, Canary, Rolling updates).
- **Benefits:**
 - Rapid delivery of new features and fixes.
 - Eliminates manual deployment errors.
 - Maximizes business agility.

Infrastructure Automation

CI/CD for Infrastructure: Why Automate Your Network?

- **Problem:** Manual network changes are slow, error-prone, and lack traceability.
- **Solution:** Apply CI/CD principles to network configuration.
- **Key Benefits:**
 - **Version Control:** Track every configuration change.
 - **Validation:** Test changes *before* deployment.
 - **Rollback:** Quickly revert to a known good state if issues arise.
 - **Auditability:** Historical insight into all network modifications.
 - **Integration:** Seamlessly connect with change and ticketing systems.

CI/CD for Infrastructure: How It Works

- **Continuous Integration (CI):**

- Store all network configurations in version-controlled repositories (e.g., Git).
- Automated checks (linting, syntax validation) on every commit.

- **Continuous Delivery (CD):**

- Verify changes in a virtual environment (e.g., Cisco Modeling Labs - CML).
- Create a "digital twin" of your production infrastructure for safe testing.

- **Continuous Deployment (CD):**

- Automate configuration deployment to the production network.
- Integrate with monitoring and ticketing systems for feedback and control.

GitLab Deep Dive

GitLab: Your Complete DevOps Platform

- **Unified Solution:** Project management, Git repositories, and built-in CI/CD in one platform.
- **Key Capabilities:**
 - Code & Configuration Management
 - Issue Tracking & Collaboration
 - Automated Pipelines & Deployments
 - Security Scanning & Monitoring

GitLab: Project & Repository Management

- **Projects:** Organize code, configurations, documentation, and wikis.
 - Create from scratch, templates, or import existing.
 - Configurable visibility: Private, Internal, Public.
- **Files:** View, edit, and create files directly within your project repository.
- **Commits:** Track and manage all changes to your project.
- **Branches:** Isolate development efforts, facilitate code review via Merge Requests.

GitLab: Collaboration & Workflow

- **Issues:**
 - Collaborate on ideas and plan work.
 - Track status of tasks and proposals.
- **Merge Requests (MRs):**
 - Propose changes from one branch to another.
 - Enable inline code review and discussion before merging.

GitLab: Pipelines – The Heart of CI/CD

- **Fundamental Building Blocks:** Orchestrate your CI/CD workflow.
- **Components:**
 - **Jobs:** Individual tasks to be executed (e.g., run tests, deploy config).
 - **Stages:** Define the order in which jobs run (e.g., build, test, deploy).
- **Typical Infrastructure Pipeline Stages:**
 - pyats-pre-snapshot: Capture current network state.
 - deploy-configuration-change: Apply configuration changes.
 - pyats-post-snapshot: Verify new network state.

GitLab: Runners – Distributed Execution

- **Purpose:** Execute jobs defined in your CI/CD pipelines.
- **How it Works:**
 - When a pipeline triggers, a Runner clones the repository.
 - It executes instructions from the `.gitlab-ci.yml` file.
 - Sends results back to GitLab.
- **Flexibility:** Supports various executors (SSH, Shell, Docker, Kubernetes) for diverse environments.

GitLab: .gitlab-ci.yml – Pipeline Definition

- **YAML Configuration:** Defines your CI/CD pipeline steps.

```
default:
  before_script:
    - export ANSIBLE_HOST_KEY_CHECKING=False # Suppress SSH host key checking for Ansible

image: 'adrianii/cicd-en:0.1.0' # Docker image for the pipeline environment

stages:
  - pyats-pre-snapshot # Stage 1: Capture pre-change snapshot
  - deploy-OSPF # Stage 2: Deploy OSPF configuration
  - pyats-post-snapshot # Stage 3: Capture post-change snapshot

pyats-pre-snapshot:
  stage: pyats-pre-snapshot
  script: cd pyats && pyats run job job.py --testbed-file test_env.yaml --trigger-datafile pre_trigger_datafile.yaml --html-logs pre_snapshots

deploy-OSPF:
  stage: deploy-OSPF
  script: ansible-playbook -i hosts actions/configure-ospf/configure-ospf.yaml

pyats-post-snapshot:
  stage: pyats-post-snapshot
  script: cd pyats && pyats run job job.py --testbed-file test_env.yaml --trigger-datafile post_trigger_datafile.yaml --html-logs post_snapshots
```

GitLab: Advanced Pipeline Features

- **Schedules:**

- Run pipelines at predefined intervals (e.g., daily, weekly).
- Ideal for compliance checks, routine audits, or scheduled deployments.

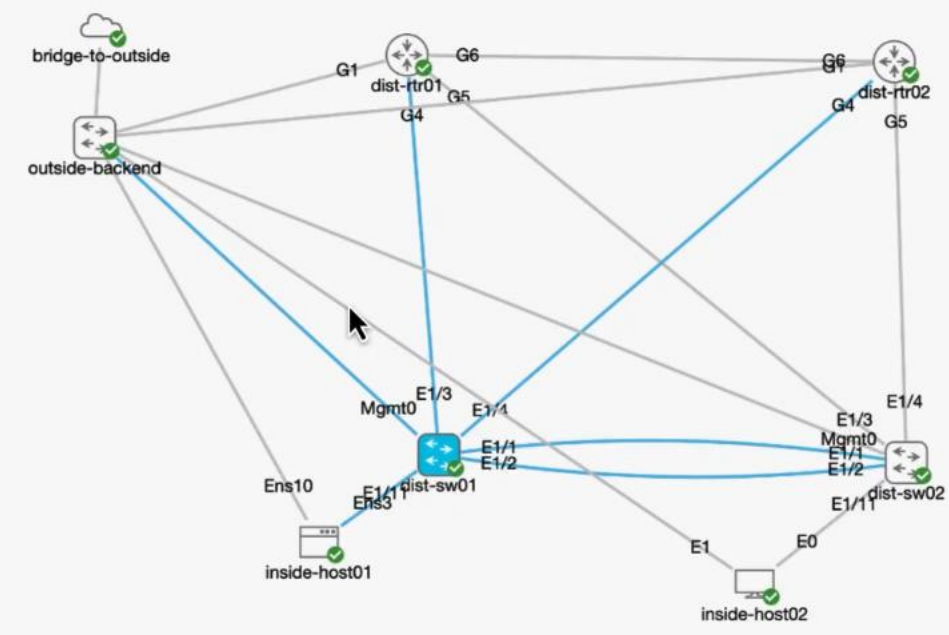
- **Environment Variables:**

- Customize job execution without modifying the `.gitlab-ci.yml` file.
- Securely inject sensitive information (e.g., API keys, credentials).

Demo

Test Environment

- **Hosting Server:** Where GitLab and other tools reside.
- **Dev Workstation:** Developer's interface for code changes.
- **Source Control (GitLab):** Version management for all configurations.
- **Infrastructure as Code (Ansible/Terraform):** Defines desired network state.
- **Network Simulation (Cisco Modeling Labs - CML):** Digital twin for safe testing.
- **Testing Framework (pyATS):** Automates network state verification.



Conclusion

Challenges in Adopting CI/CD for Networks

- **Cultural Shift:** Moving from manual CLI to code-driven automation.
- **Skill Gap:** Need for network engineers with programming and DevOps skills.
- **Tooling Complexity:** Integrating various automation, testing, and CI/CD platforms.
- **Legacy Infrastructure:** Automating older devices or proprietary systems can be difficult.
- **Testing Environment:** Creating accurate and scalable virtual labs (digital twins) for testing.
- **Security Concerns:** Ensuring automation doesn't introduce new vulnerabilities.
- **Organizational Buy-in:** Gaining support from management and other teams.

Best Practices for Network CI/CD

- **Start Small:** Automate a single, well-defined process or device type first.
- **Version Control Everything:** Treat all configurations, scripts, and playbooks as code.
- **Build a Digital Twin:** Invest in a robust network simulation environment (e.g., CML) for testing.
- **Automate Testing:** Implement comprehensive pre- and post-change validation with tools like pyATS.
- **Embrace Collaboration:** Foster a DevOps culture between network and software teams.
- **Secure Your Pipeline:** Implement access controls, secrets management, and security scanning.
- **Iterate and Improve:** Continuously refine your automation and CI/CD processes.
- **Document Thoroughly:** Maintain clear documentation for all automated workflows.

Resources

Resources & Next Steps

- **Cisco Blogs:** <https://blogs.cisco.com/author/adrianiliesiu>
- **GitHub Repository:** <https://github.com/aidevnet/cicd-twitch>
- **YouTube Playlist:** <https://www.youtube.com/playlist?list=PL2k86RIAekM-Vv4pLLFYsQ6Re9BBlj4dt>
- **NetDevOps Learning:** <https://netascode.cisco.com/>

Complete your session evaluations



Complete a minimum of 4 session surveys and the Overall Event Survey to claim a Cisco Live T-Shirt.



Earn up to 800 points by completing all surveys and climb the Cisco Live Challenge leaderboard.



Level up and earn exclusive prizes!



Complete your surveys in the Cisco Live Events app.

Continue your education



Visit the Cisco Stand for related demos



Book your one-on-one Meet the Expert meeting



Attend the interactive education with Capture the Flag, and Walk-in Labs



Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand

Thank you

CISCO Live !

