

# <Code is the new CLI />

Network Programmability and beyond

Alfonso Sandoval Rosas  
Software Consulting Engineer

November 2023

# Who is this?

Software Consulting Engineer

SWAT (Software & Automation Team) Lisbon 

[github.com/ponchotitlan](https://github.com/ponchotitlan)

[linkedin.com/in/asandovalros](https://linkedin.com/in/asandovalros)





Spreadsheet-based



Manual CLI copy/paste



Lots of time and rework  
due to errors



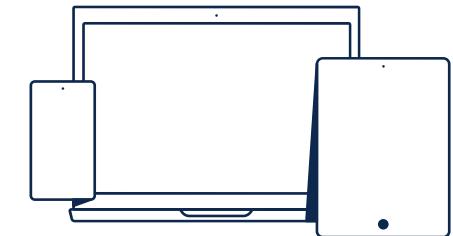
Setup in a multivendor  
network



Operations Team



LDAP-based secure  
authentication

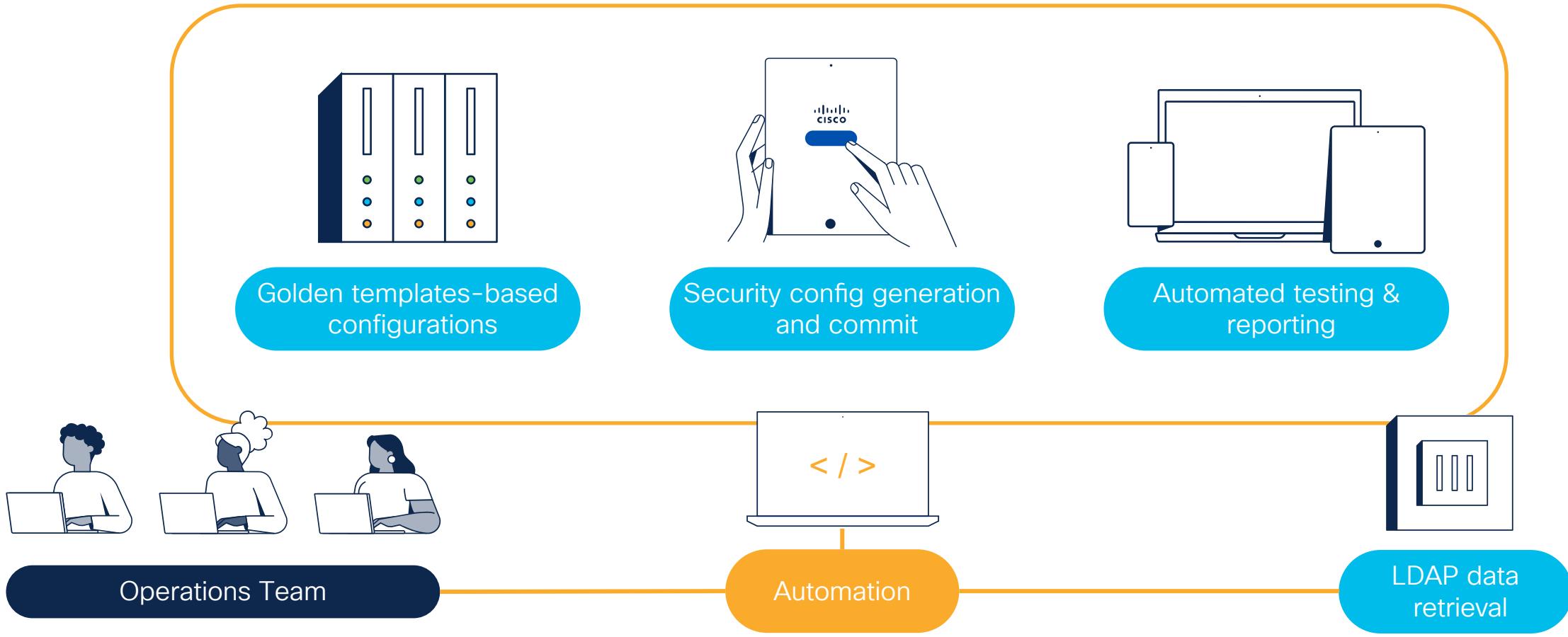


Testing and  
troubleshooting

# But first, a horror story ...

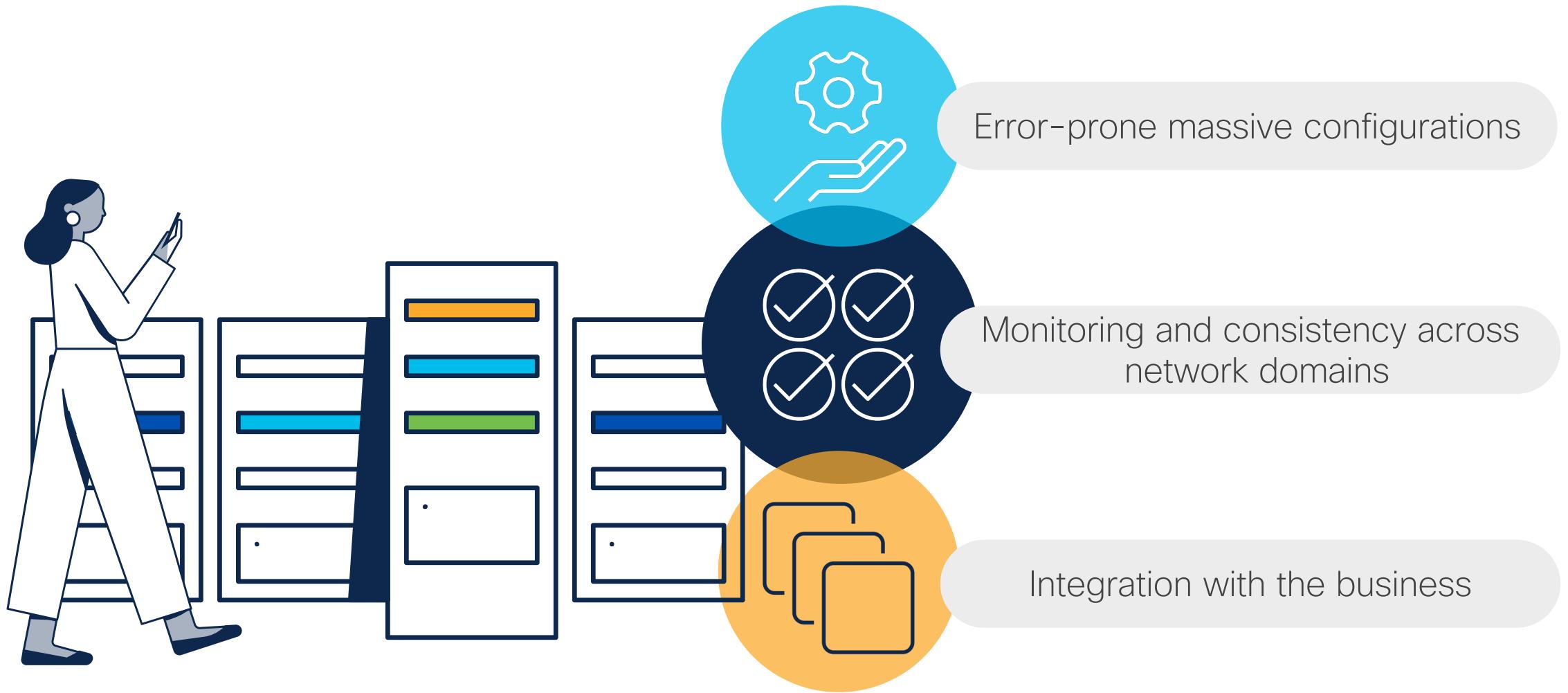
Enable massive WFH bank trading amid COVID in record time! ⏳





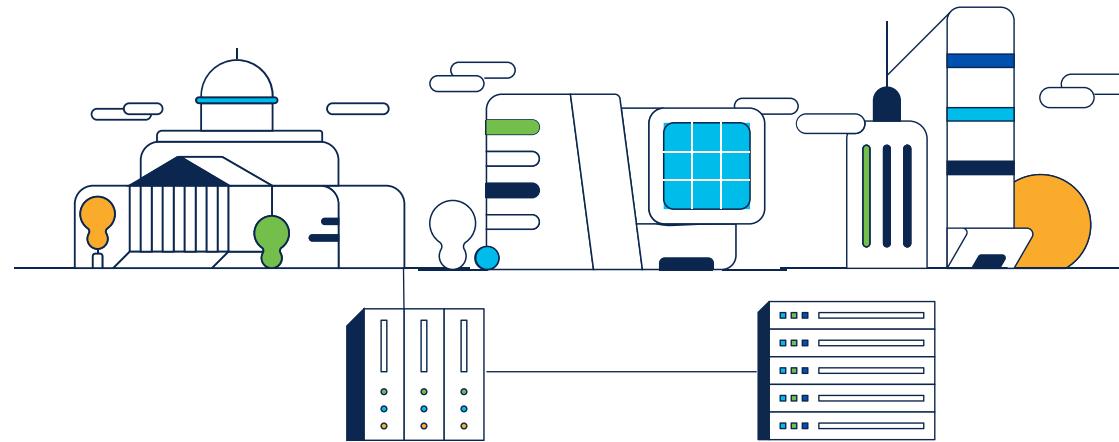
But first, a horror story ...  
Enable massive WFH bank trading amid COVID in record time!

# What tends to go wrong



# Current challenges in our networks

**95%**  
of the changes in the  
network are done  
manually



**70%**  
of policy violations are  
caused by human  
intervention

**75%**  
of the OpEx is invested on  
troubleshooting and visibility  
of the network

\*Study performed by Gartner Inc. 2022 in USA



# Code is the new CLI

# Today's agenda



Managing the CLI with code



Model-driven programmability



IaC (Infrastructure as Code)



NetDevOps

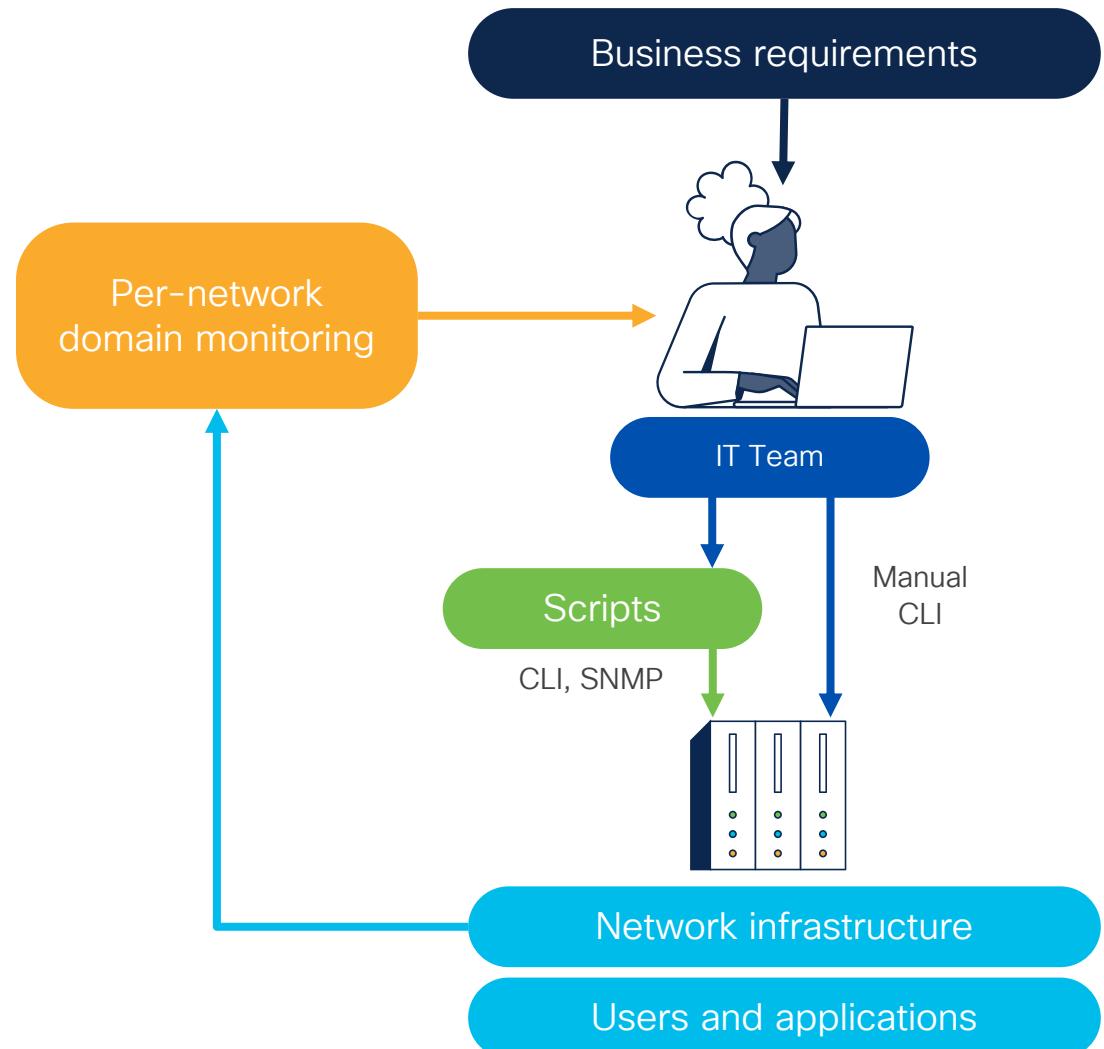


Career tips & tricks

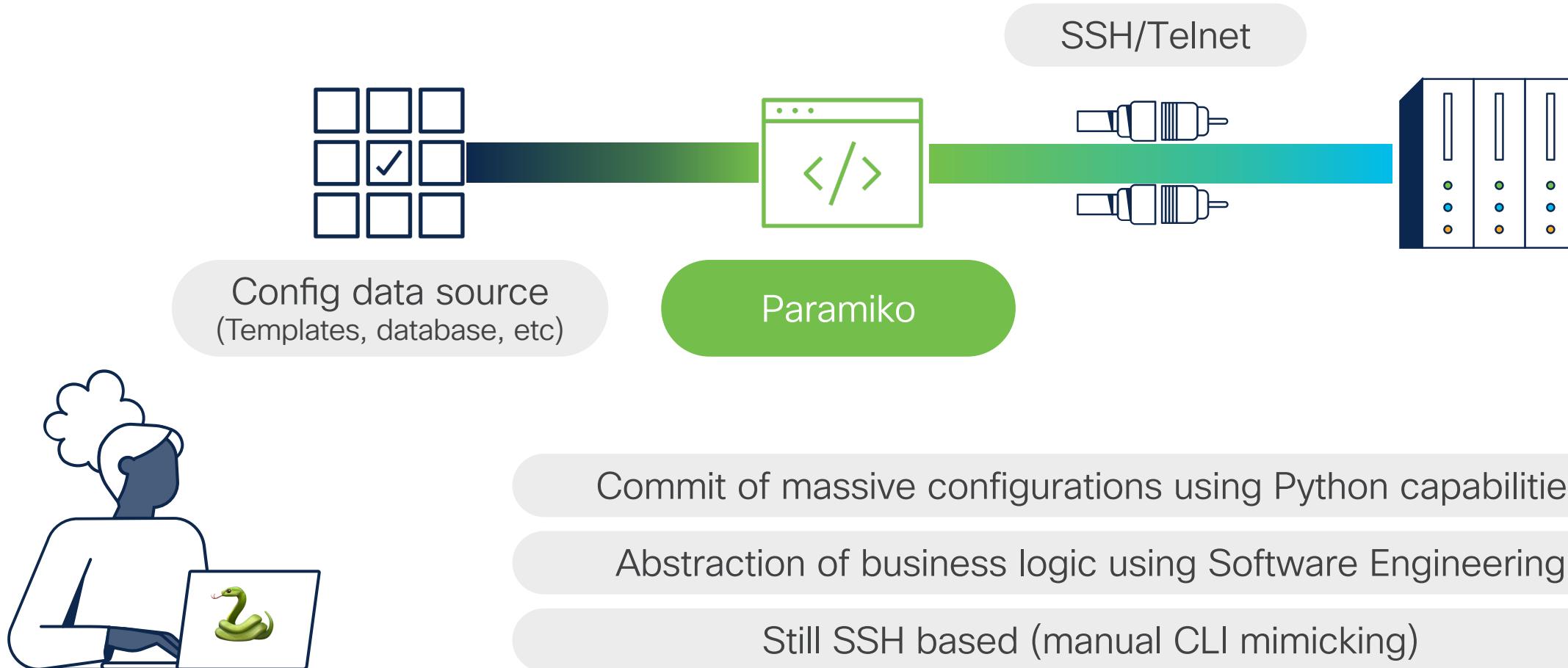


## Managing the CLI with code

- Focus on highly recurring tasks
- Per box configuration
- Mimicking of CLI interaction
- **CLI is designed to be used by humans and not for a programmatic access!**



# Evolving scripts with 🐍 Python



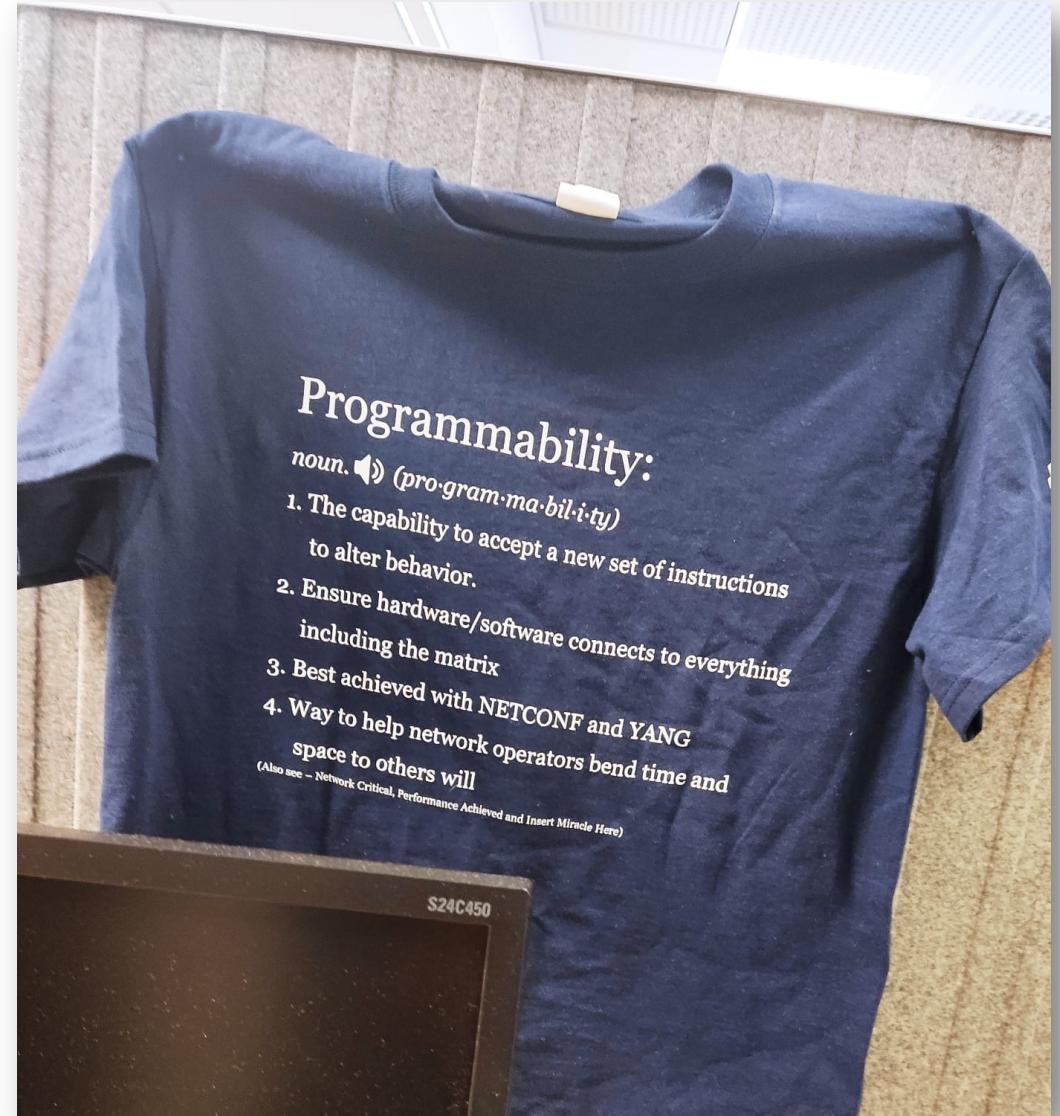


# Demo CLI interaction with Python Paramiko



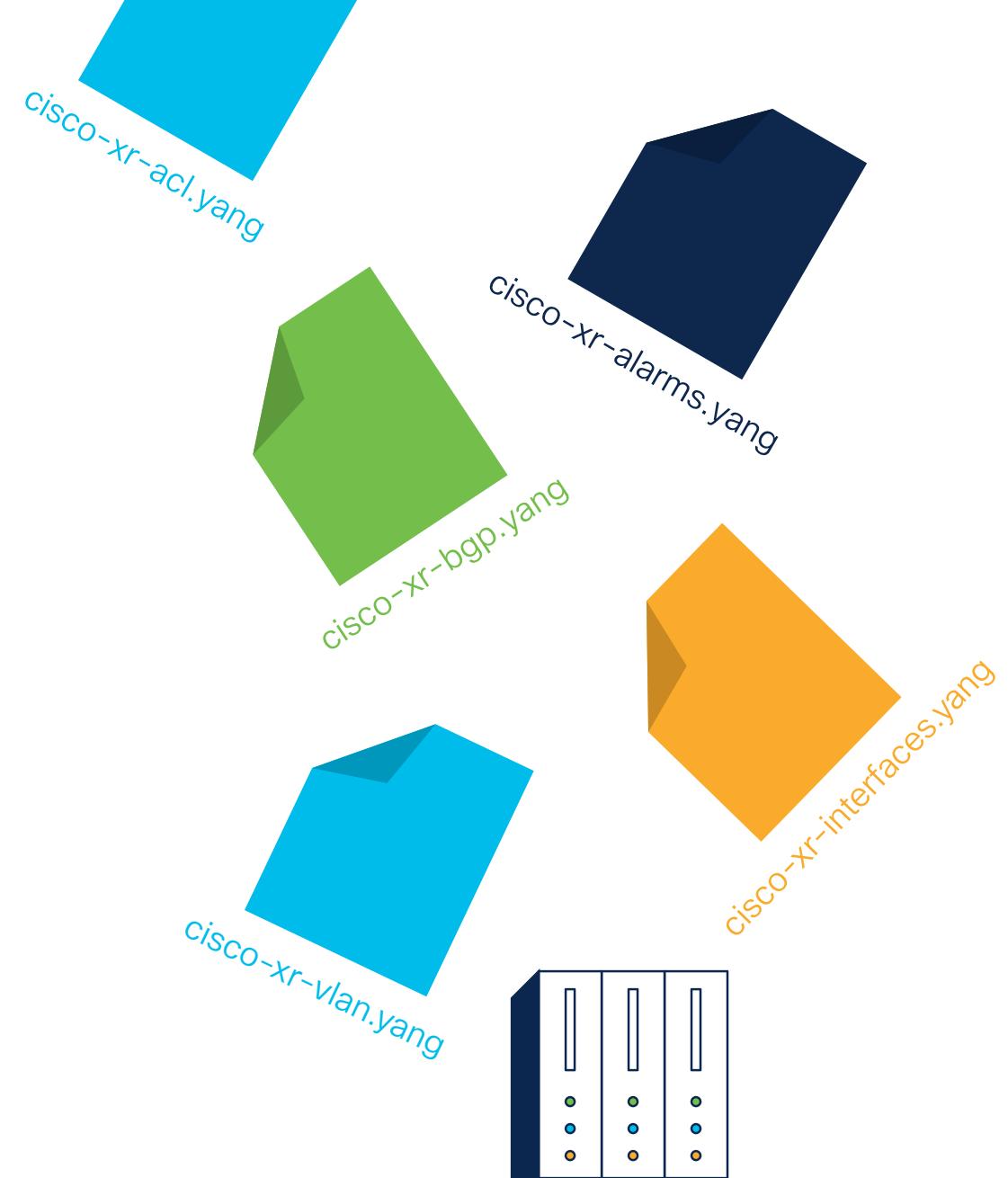
# Model-driven programmability

- Going beyond mimicking CLI or operating with SNMP
- In-built mechanisms for altering network devices behavior using protocols and standards
- IETF (Internet Engineering Task Force) RFCs



# YANG data models

- RFC 6020
- Data modelling language
- Models configurations and state data of a device or service
- Organized in nodes
- Plenty of node and data types
- **Device Data Models** (Interface, VLAN, etc)
- **Service Data Models** (L3VPN, VRF, etc)
- **Industry Standard** vs. Vendor Specific



# YANG data models

Leaf 

```
leaf host-name {  
    type string;  
    description "Hostname for this system";  
}
```

Simple data like integer or string. It holds exactly one value of a type, and has no children

# YANG data models

Leaf-list 

```
leaf-list domain-search {  
    type string;  
    description "List of domain names to search";  
}
```

Sequence of leaf nodes with exactly one value of a particular type per leaf

# YANG data models

Container 

```
container login {  
    leaf message {  
        type string;  
        description  
            "Message given at start of login session";  
    }  
}
```

Grouping of related nodes in a subtree. It has only child nodes and no value.  
May contain any number of child nodes of any type

# YANG data models

List 

```
list interface {
    tailf:info "Assign interface as port0";
    key name;
    leaf name {
        type string {
            tailf:info "Interface Name";
    }
    . . .
}
```

Sequence of entries uniquely identified by the value on their key leaf.  
A list can define multiple keys and may contain any number of child nodes of any type



IT Team

# NETCONF protocol

- RFC 6241
- Commit, edit and delete configurations on network devices
- Successor of SNMP, but for config and monitoring
- Based on XML for encoding
- Comms via RPCs (Remote Procedure Calls)
- SSH based, port no. 830 as default
- YANG models are used to operate the device's config

Content  
Configuration/Notification data

Operations

```
<edit-config> <copy-config> <delete-config> ...
<get> <get-schema> <get-data> ...
```

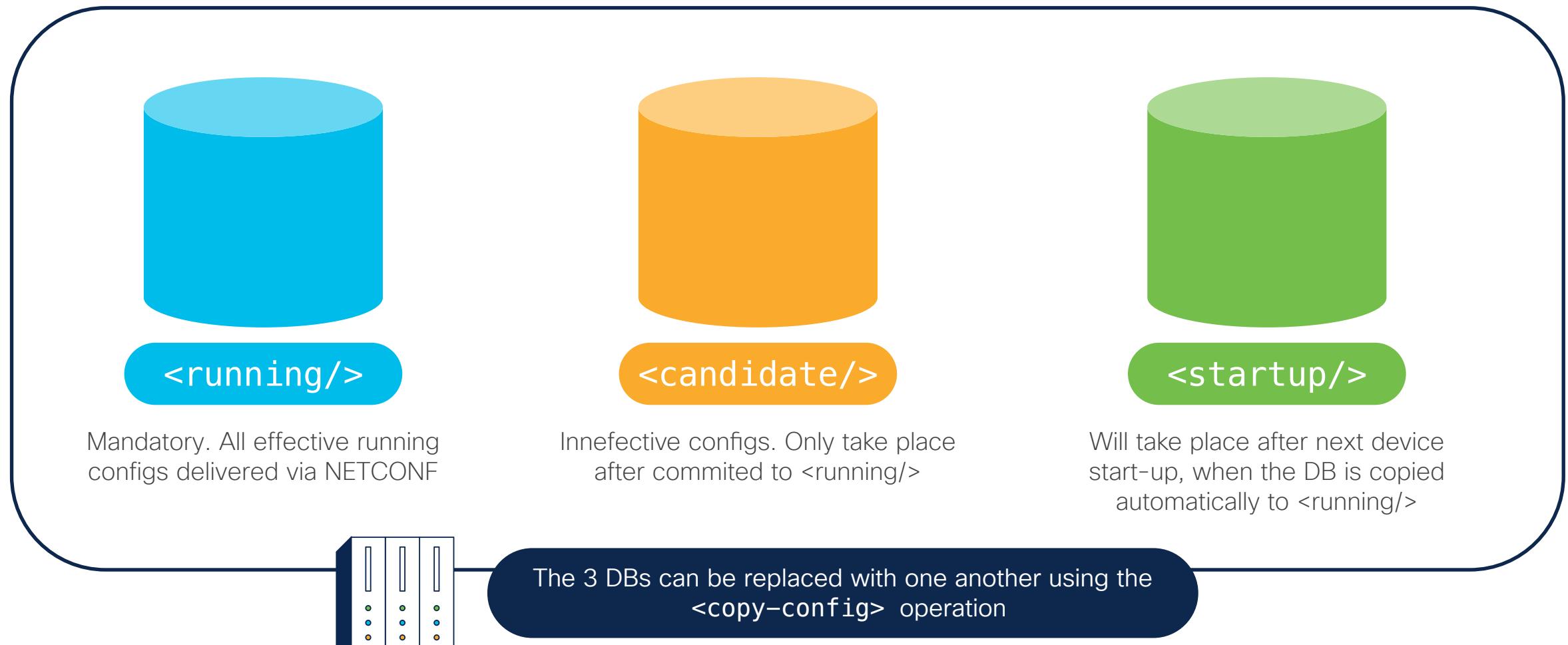
Messages

```
<rpc> <rpc-reply> <notification>
```

Secure Transport  
SSH, TLS



# NETCONF protocol



# NETCONF protocol

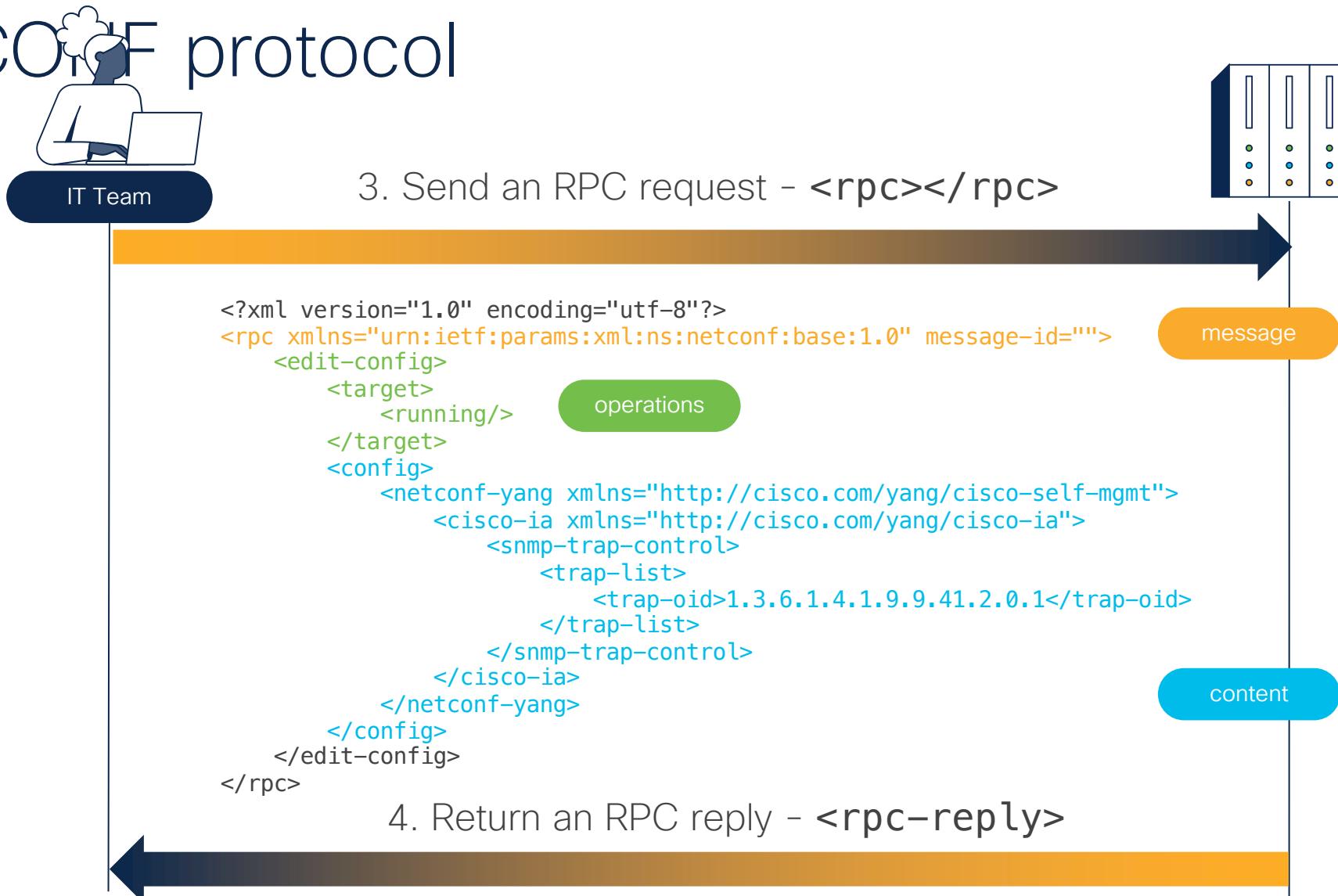


1. Establish an SSH connection

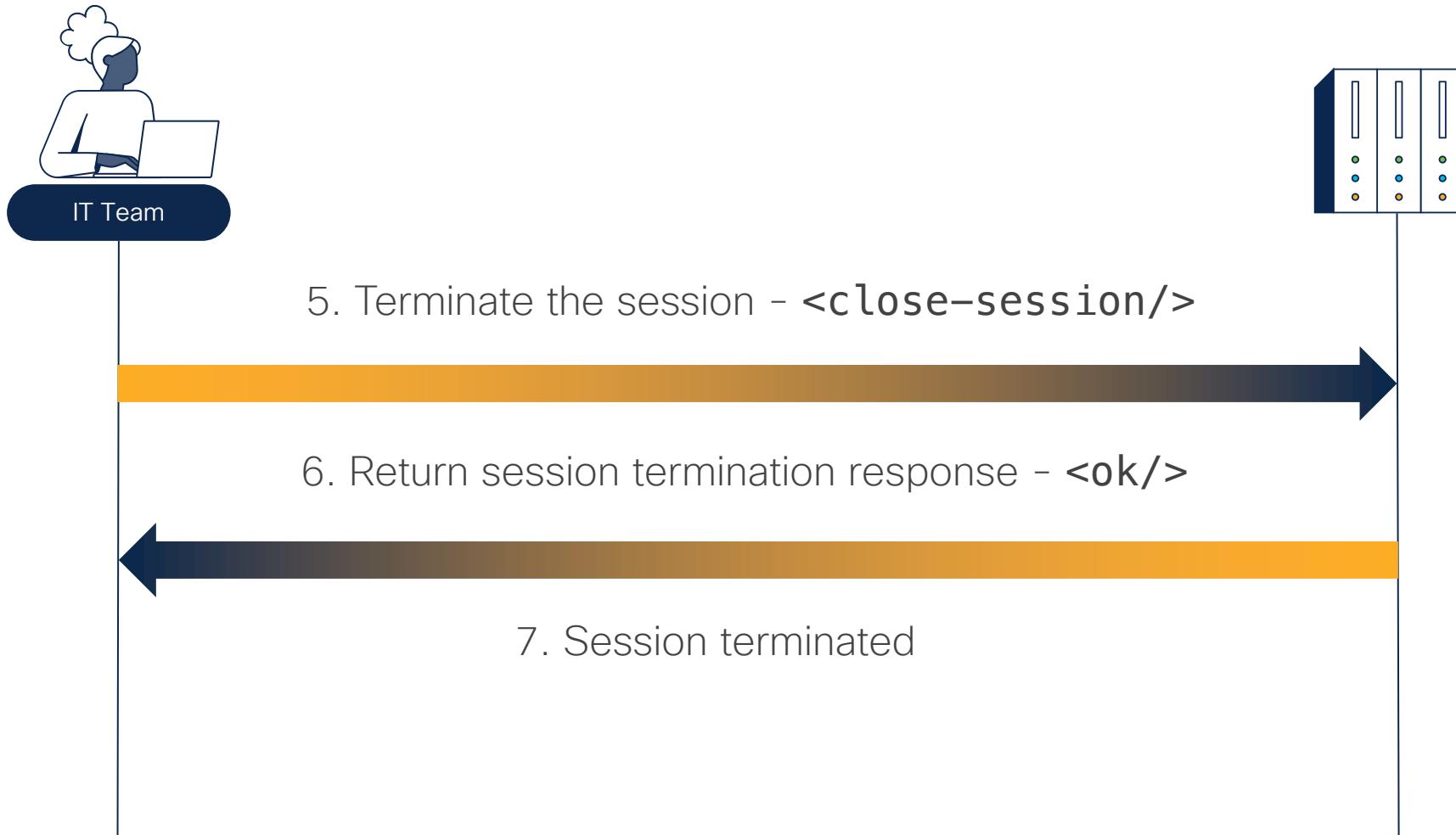
2. Send **hello** messages to advertise supported capabilities

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability> ... :base:1.0</capability>
    <capability> ... :base:1.1</capability>
    <capability> ... :writable-running:1.0</capability>
    <capability> ... :xpath:1.0</capability> ...
```

# NETCONF protocol



# NETCONF protocol

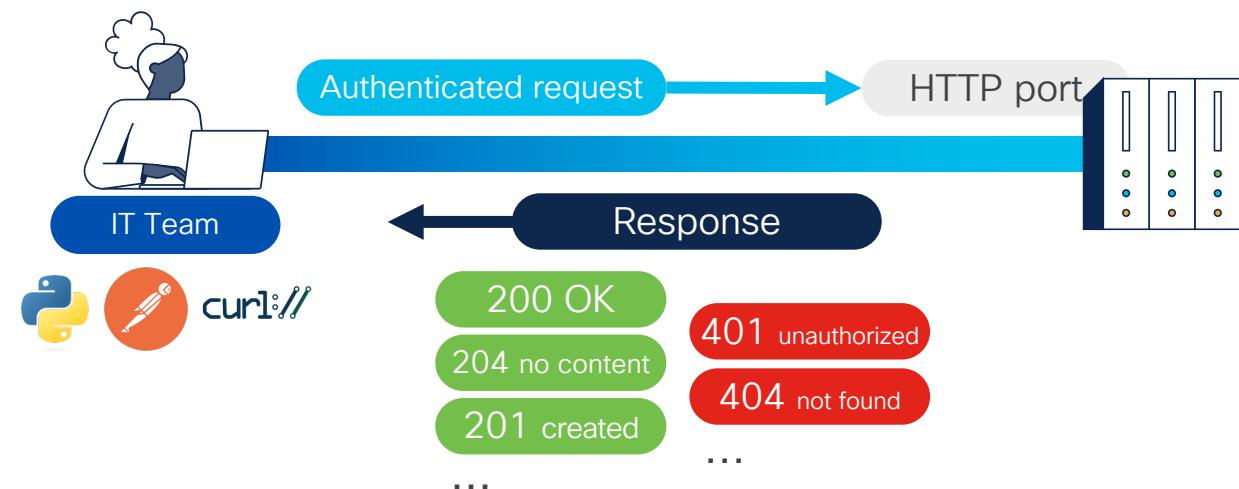
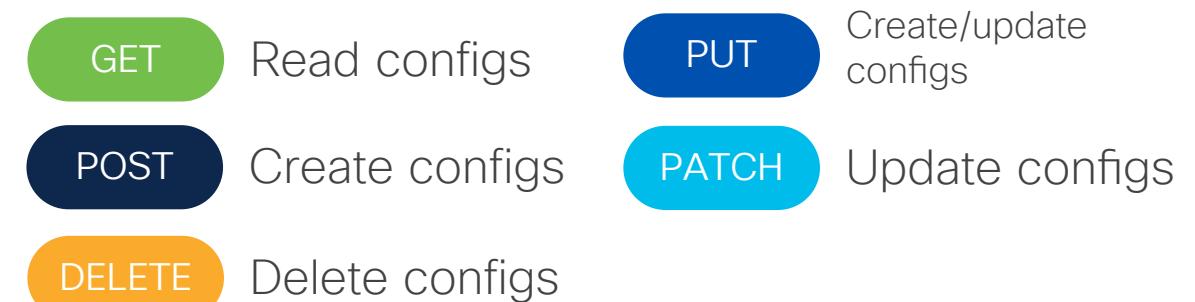




# Demo NETCONF & YANG with Python ncclient

# Do you remember REST?

- HTTPS-based communications
- Stateless
- RESTful API interface for operations
- Verbs for CRUD actions
- Standardized response codes



# RESTCONF protocol

- RFC 8040
- Based on XML or JSON for data encoding
- RESTful-style interaction with network devices
- Also based on YANG data models



IT Team

Content (Config/Operational Data)  
XML, JSON

Operations

GET, POST, PUT, PATCH, DELETE

Transport  
HTTPS



# RESTCONF protocol URLs

`https://<ADDRESS>/<ROOT>/DATA/<[YANG MODULE:] CONTAINER>/<LEAF> [?<OPTIONS>]`

`https://<ADDRESS>/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet1?depth=unbounded`

```
module: ietf-interfaces
+--rw interfaces
| +--rw interface* [name]
| | +--rw name
| | +--rw description?
| | +--rw type
| | +--rw enabled?
| | +--rw link-up-down-trap-enable?
```

string	
string	
identityref	
boolean	
enumeration	

Ex. return all nested models until the final node is reached

Other options include:

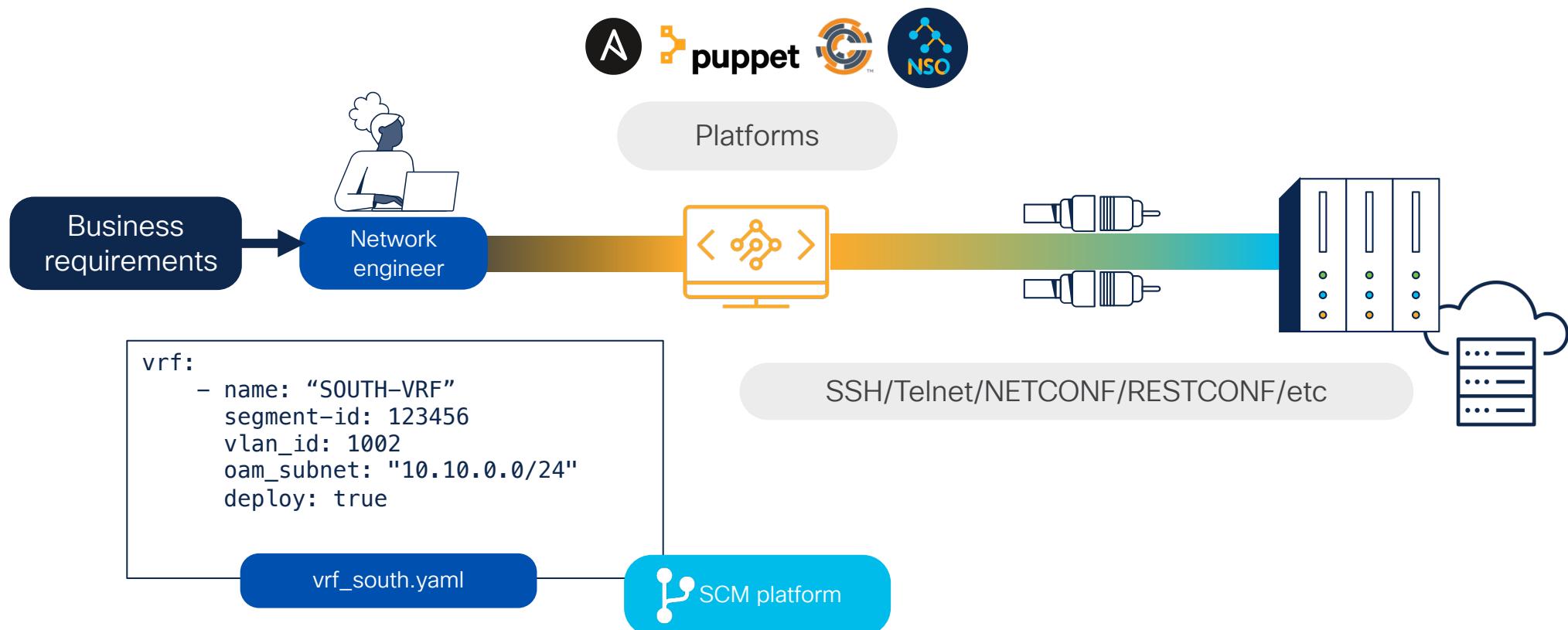
`content = [ all, config, nonconfig ]`  
(Type of data to be returned)

`fields = expression`  
(Limit which fields are returned)



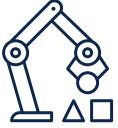
# IaC (Infrastructure as Code)

- Defining the desired state of a network using simple text files
- IaC platforms for different purposes
- Vendor plugins specific for their network devices

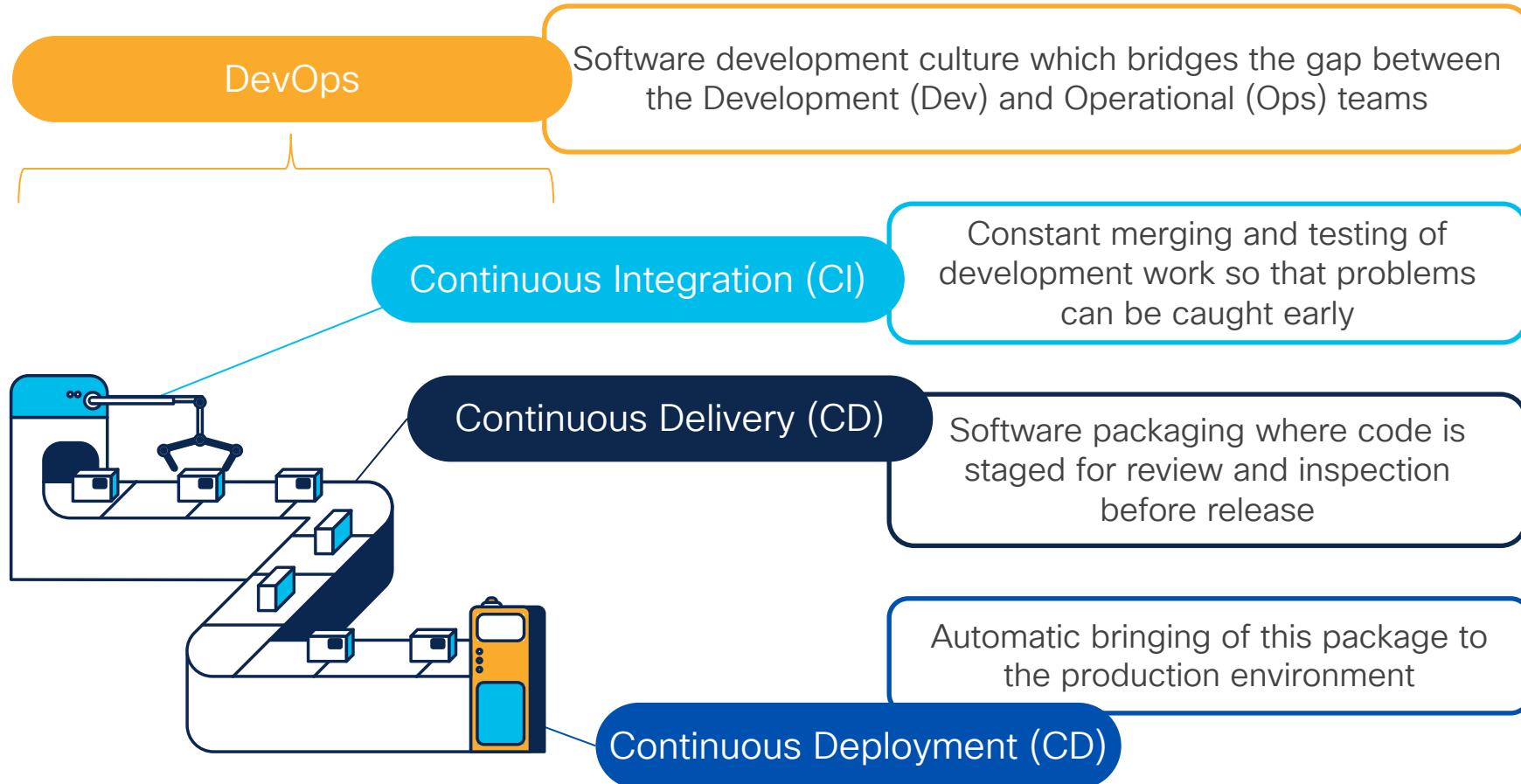




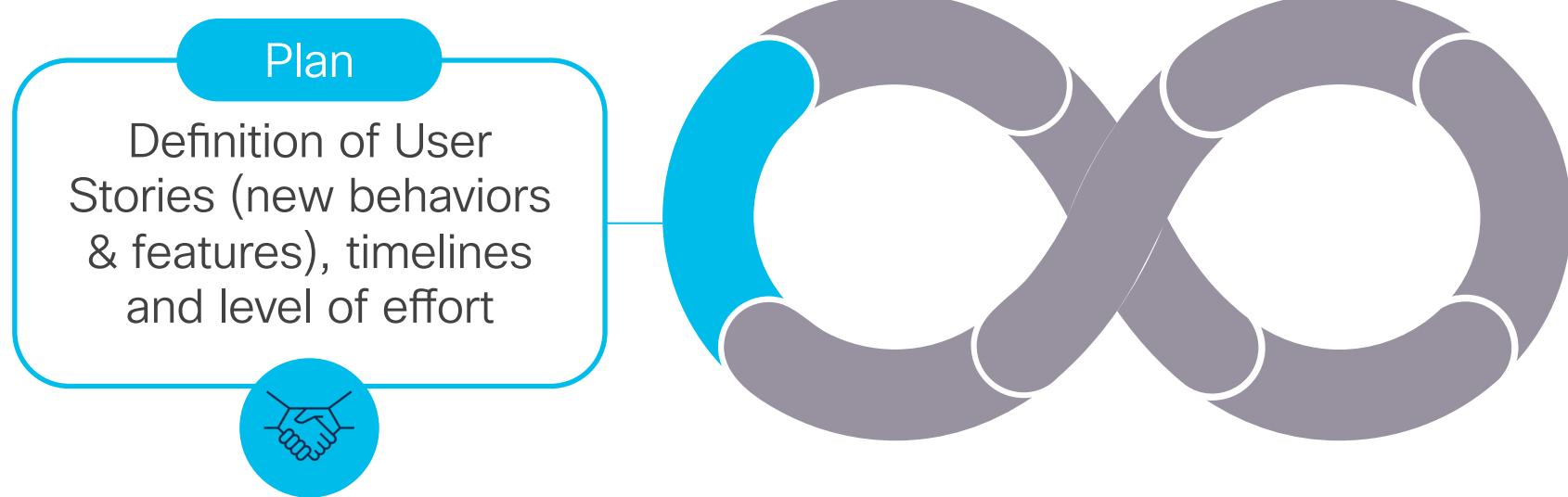
# Demo IaC with Ansible



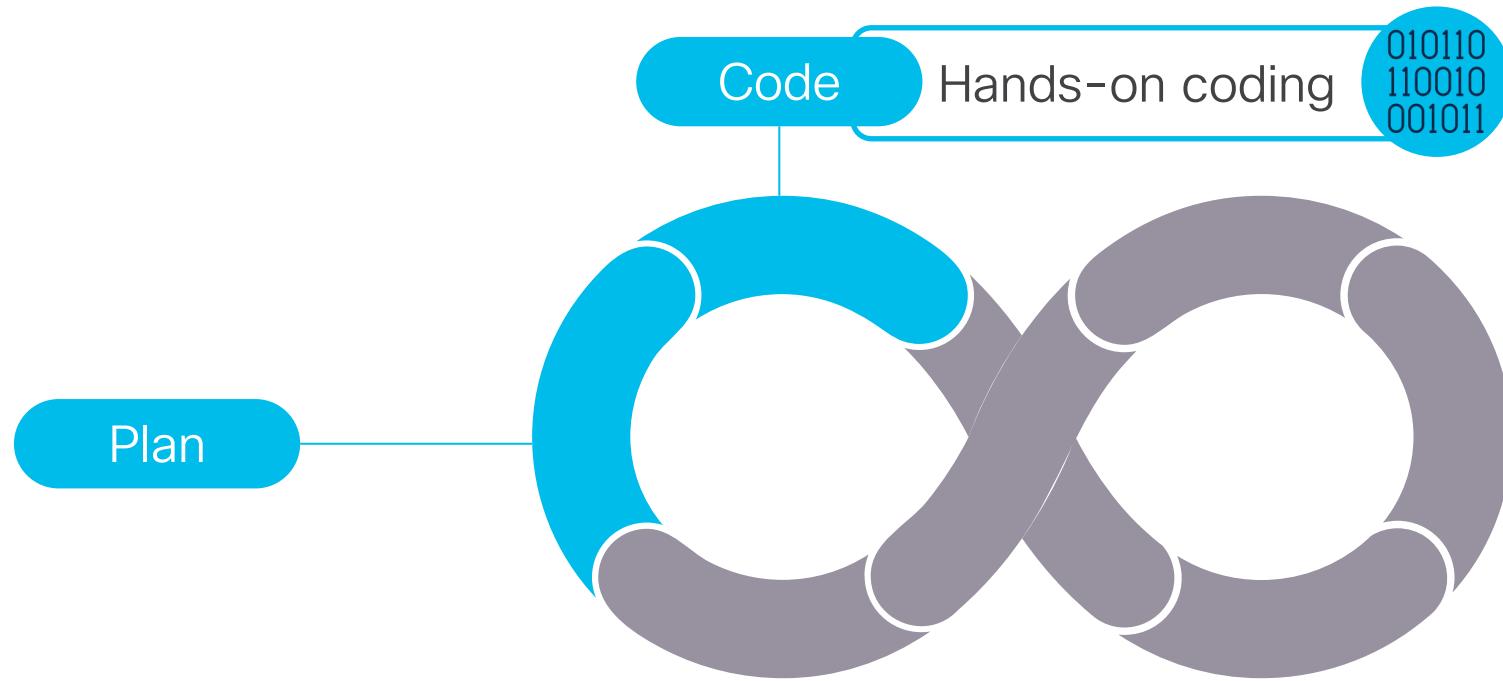
# NetDevOps



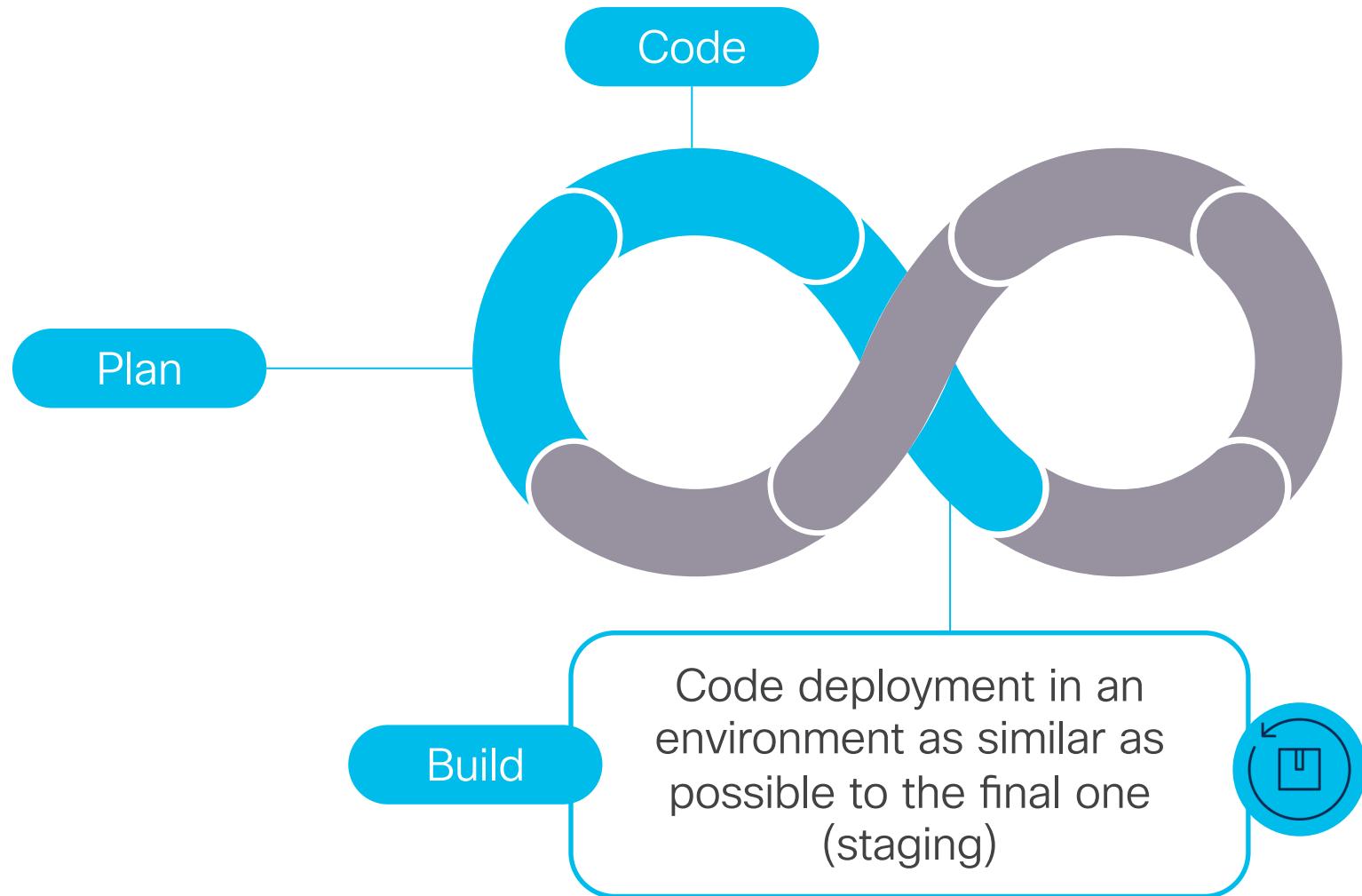
# Continuous Integration (CI) most common stages



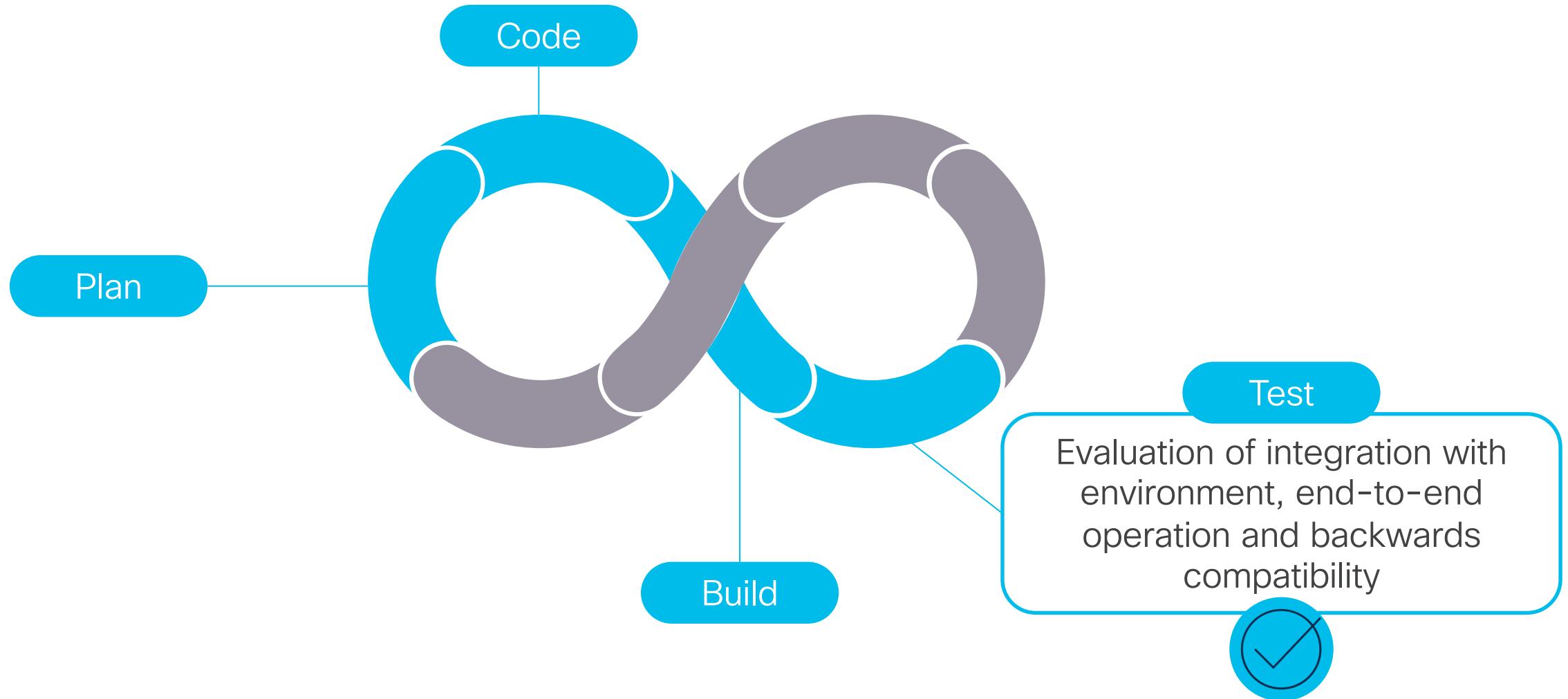
# Continuous Integration (CI) most common stages



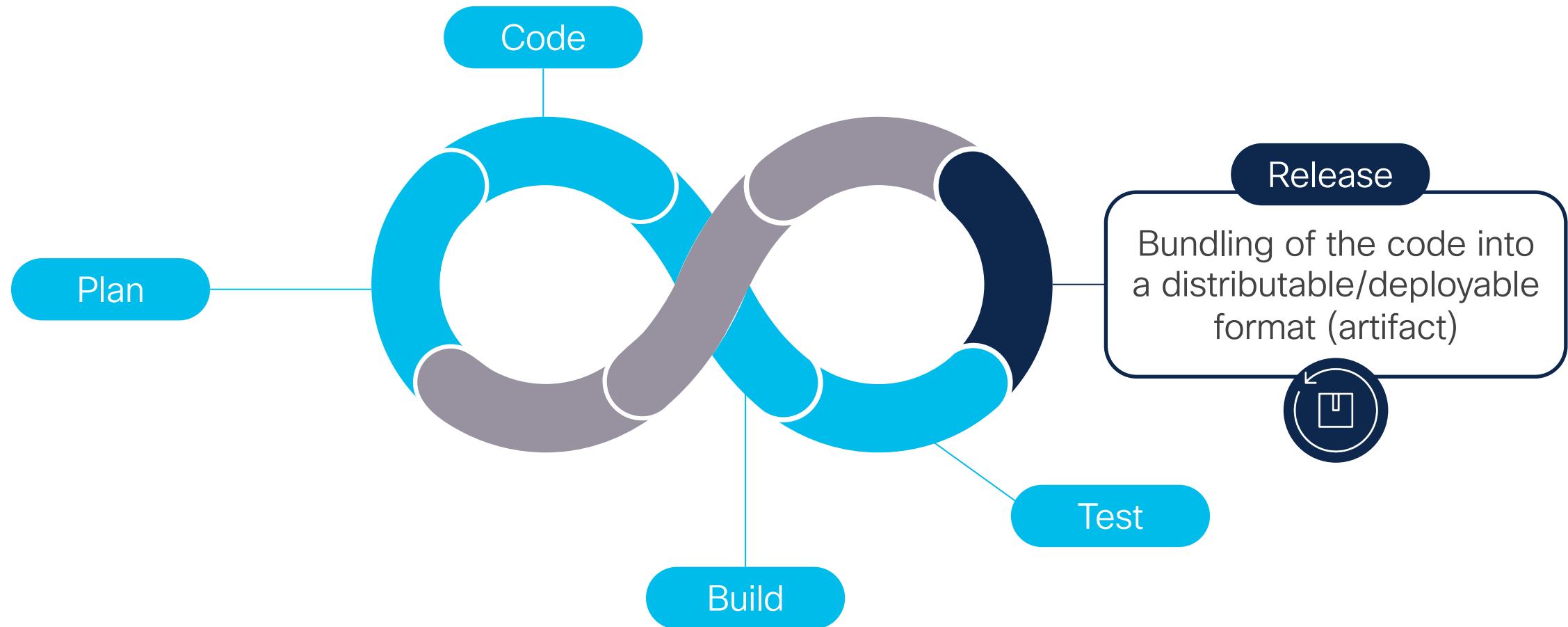
# Continuous Integration (CI) most common stages



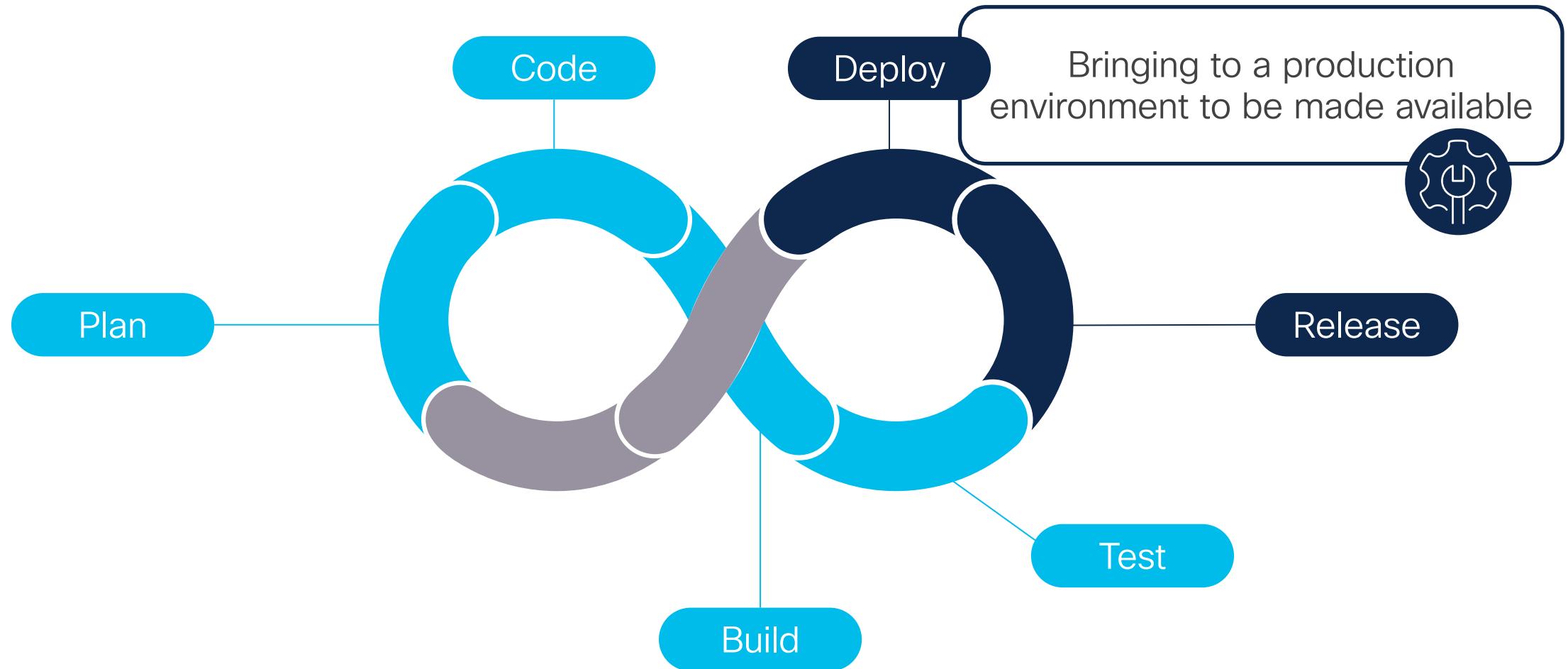
# Continuous Integration (CI) most common stages



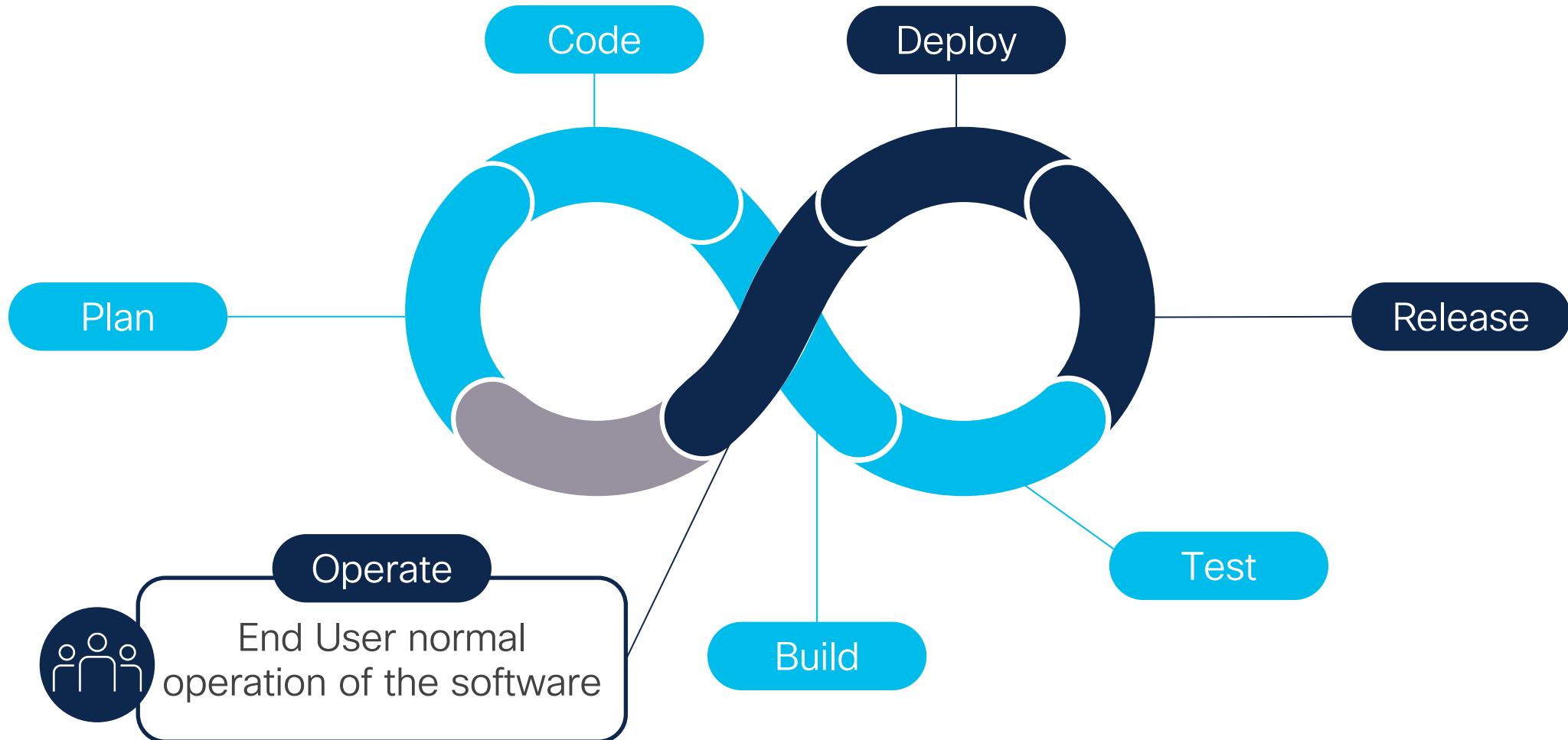
# Continuous Delivery/Deployment (CD)



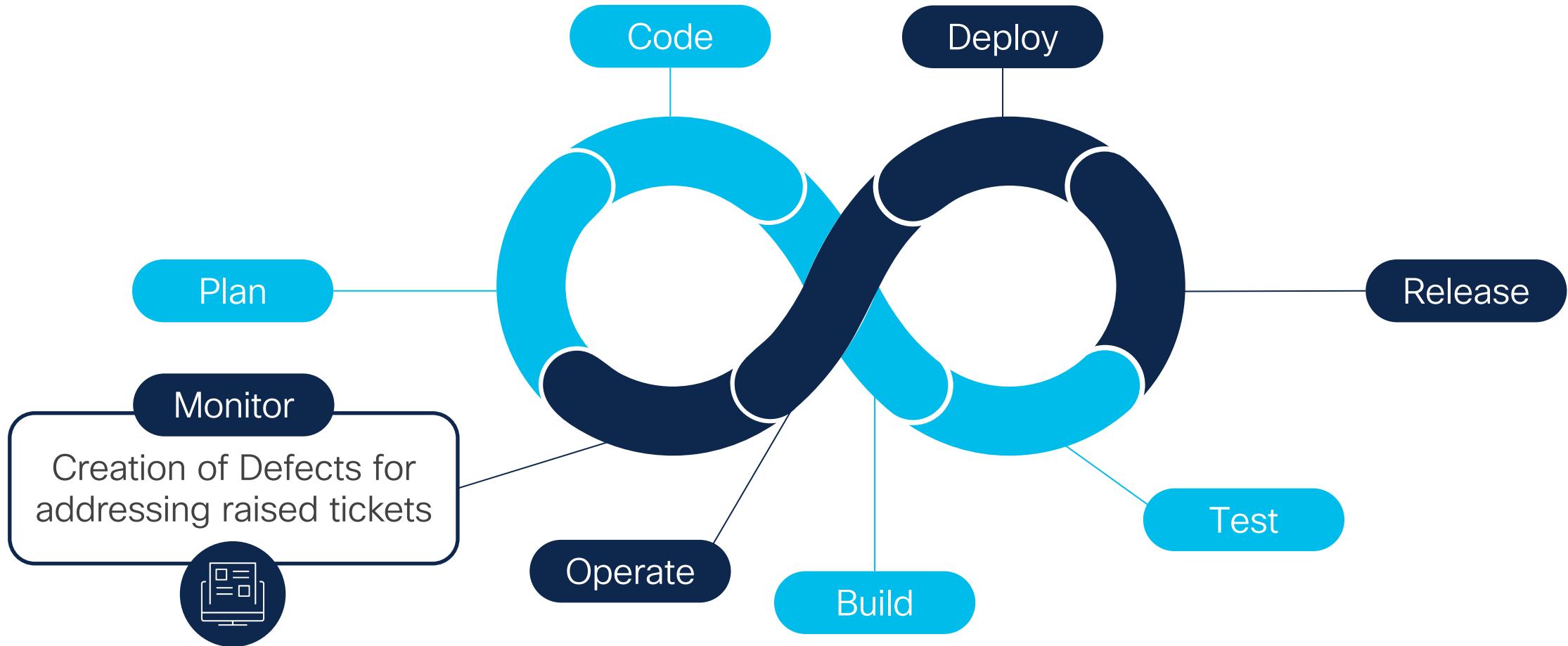
# Continuous Delivery/Deployment (CD)



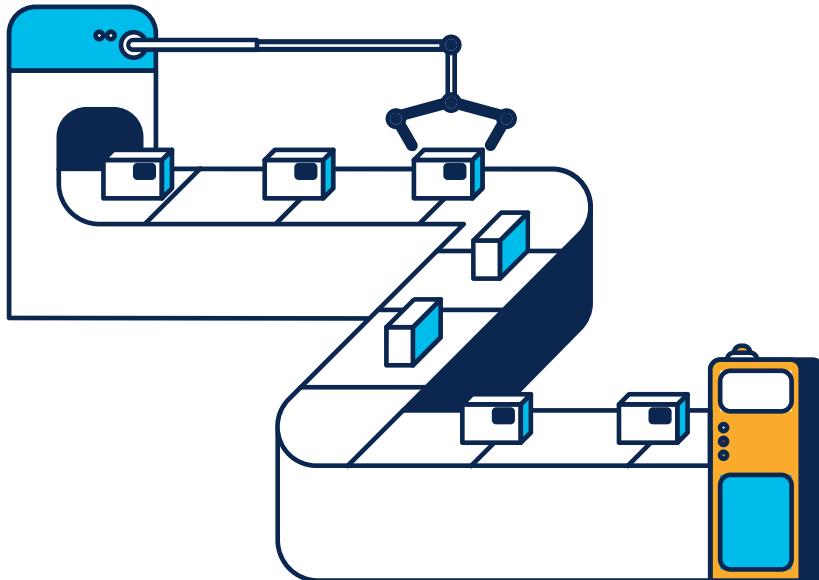
# Continuous Delivery/Deployment (CD)



# Continuous Delivery/Deployment (CD)



# Key benefits



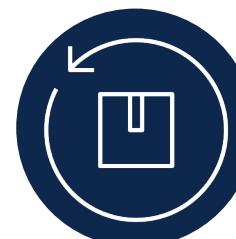
Early detection of defects by working with smaller changes



No error risk due to manual work

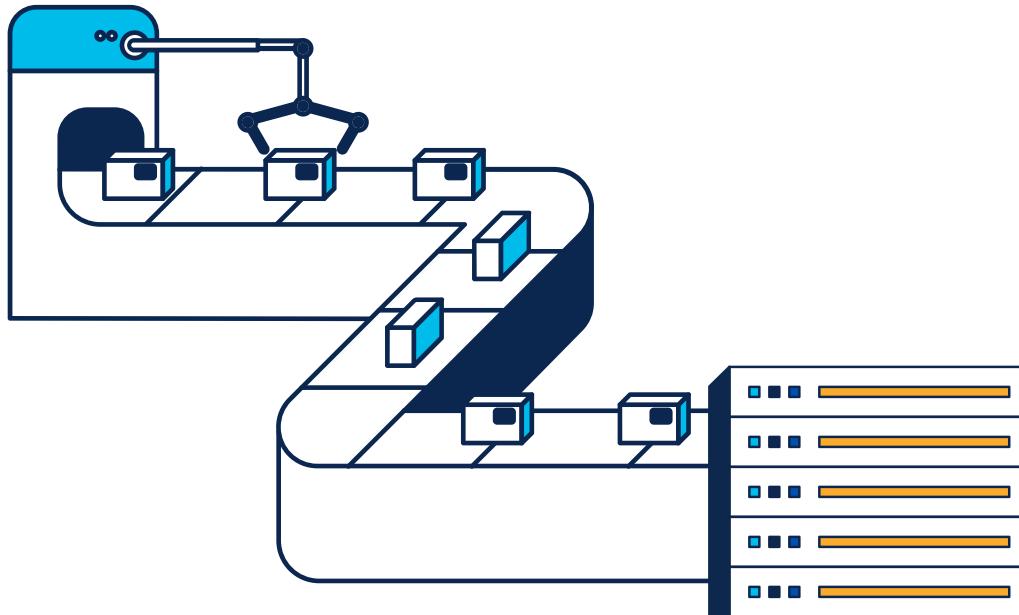


Delivery quality and reliability assurance

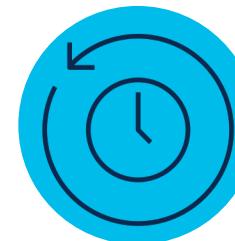


Faster, more efficient software delivery

# Networks + DevOps practices = NetDevOps



Small but frequent changes in the network services

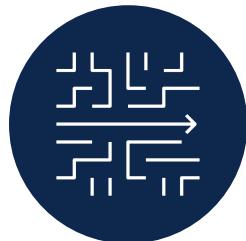


Roll service updates with more reliability

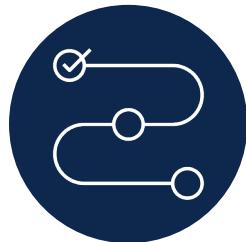


Mitigate disruption risks caused by bad code or manual intervention

# What is a pipeline?



Series of processes that run when triggered by an action



The processes are defined in stages, which are executed sequentially





# Career tips & tricks



## Software Engineering

- Solid coding understanding
- Design patterns
- OOP
- Documentation habits
- Code versioning (Git)
- Unit testing
- Python is usually the entry programming language



## Networking Engineering

- Routing & Switching concepts
- Device CLI management
- Cross-tech concepts (Enterprise, Core, SP, etc)

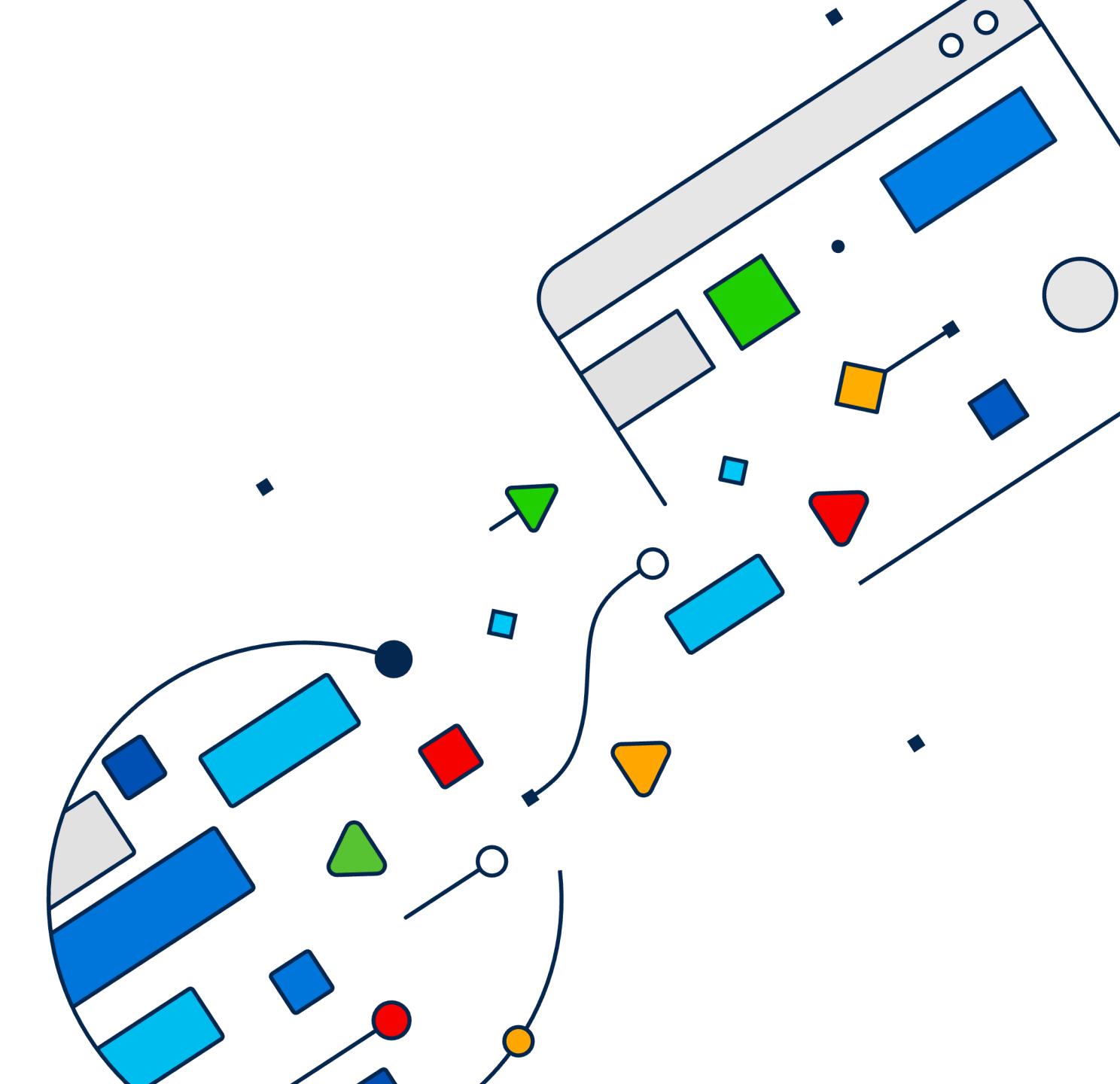


## Services & Cloud

- Containerized Technologies and Design (Docker, K8s, etc)
- Cloud Providers
- Infrastructure management
- DevOps platforms



QA





[https://github.com/ponchotitlan/  
NOVA\\_code\\_is\\_the\\_new\\_CLI](https://github.com/ponchotitlan/NOVA_code_is_the_new_CLI)



The bridge to possible