How many environments do we need?

MUST have one environment

Pros

- Lowest cost
- Lowest complexity

Cons

- No fault tolerance
- No non production place to conduct development work.
- No non production place to conduct testing.
  - Integration partners
  - Code promotion practices

Using RFC 2119 in contemplating this topic.

SHOULD consider more than one environment

Pros

- A **Development environment** is used by developers to build features for coding and testing without affecting the organization's production environment.
  - A safe place for developers to work
  - A safe place for platform administrators to learn about the platform
  - A safe place to develop integrations
- A **Testing environment** is used by an organization for quality assurance and user acceptance testing. It is used to validate changes before they go live.
  - A safe place to test code promotion processes
  - A safe place for integration partners to test integrations
- A **Staging environment** is a mirror of production that is used for final testing before deployment to production.
- A **Training environment** is useful in larger organizations for training users.

Cons

- More costs
- More complexity

SHOULD consider using Configuration as Code

Pros

- AAP configurations are in a software repository, making them highly portable and version controlled.
- Can be used to hydrate more than one AAP.
- A repeatable process for populating your AAPs.
- Much faster than populating AAP via mouse and keyboard.

Cons

- More complex

Resources

- Demo: Configuration as Code for Ansible Automation Platform - video
- Manage automation controller Configuration as Code (CaC) with Ansible - gitops
- Configuration as Code - Red Hat documentation
- Red Hat collection for configuration as code - Red Hat collection
- Config as code applied to sales demos - working code

How many **Production** environments do we need?

SHOULD consider

Recovery Time Objective (RTO)

- **Definition:** The duration of time within which business processes and IT systems **MUST** be restored after a disruption.
- **Focus:** How fast do we need to be back up?
- **Example:** A 4-hour RTO means systems **MUST** be back online within 4 hours of failure.
- **Impact:** Lower RTOs usually require higher, more expensive technology investments.

Recovery Point Objective (RPO)

- **Definition:** The maximum age of files or data in backup storage that must be recovered for normal operations to resume, indicating the acceptable data loss interval.
- **Focus:** How much data can we afford to lose?
- **Example:** A 1 hour RPO means that in a disaster, you can lose up to 1 hour of data.
- **Impact:** Lower RPOs require more frequent backups or synchronous replication.

SHOULD consider

Key differences between RTO & RPO

- **RTO:** Measures time from the incident forward (downtime).
- **RPO:** Measures time backward from the incident (data loss).
- **Purpose:** RTO helps define the disaster recovery strategy, whereas RPO helps define the backup strategy.

Setting your targets

Not all applications require the same level of protection. Organizations typically use a tiered approach to balance cost and resilience.

- **Tier 1 (Mission Critical):** RTO: 15-60 minutes; RPO: <5 mins (near zero).
- **Tier 2 (Business Critical):** RTO: 15-60 minutes; RPO: 15-60 mins.
- **Tier 3 (Important):** RTO/RPO: 1-4 hours.
- **Tier 4 (Low priority):** RTO/RPO: 4-24+ hours.

SHOULD consider

Resources:

- Red Hat Ansible Automation Platform
- PostgreSQL 15 Chapter 27. High Availability, Load Balancing, and Replication
- Scalability Rules: 50 Principles for Scaling Web Sites
- Ansible on Azure Articles
- Ansible Automation Platform Service on AWS Articles

Downtime calculator:

1. How much does downtime cost you?
   a. $ amount
2. How often does that happen?
   a. % of 100
3. Calculate the value

High Availability

- **Active** is running your production automation.
- **Ready** is ready to take over if needed.
- **Maintenance** is being maintained.

Active

Maintenance

Ready