

# Setting up your own Checkmk staging environment

CHECKMK CONFERENCE #6 – MUNICH, APRIL 29, 2020



**Marsellus Wallace**

Consultant

tribe29

# Agenda

1. CREATING STAGING ENVIRONMENTS
2. BEST PRACTICES FOR TESTING UPDATES



# Agenda

1. CREATING STAGING ENVIRONMENTS
2. BEST PRACTICES FOR TESTING UPDATES





# Staging environments: 2 ways

1

## **GOLD STANDARD**

Precise, simple but resource intensive

2

## **SILVER STANDARD**

Good feature coverage but lack of performance testing

# Easy to setup, but additional load

## DESCRIPTION

Staging Checkmk instance monitors productive hosts in parallel to productive Checkmk instance

## HOW IT WORKS

- Create full Checkmk staging instance in parallel to productive system by cloning all productive sites
- Tweak your slave site names in distributed monitoring

## BENEFITS

- Easy to setup
- No data modelling needed
- Most precise picture of productive environment

## DOWNSIDES

- Additional load on productive environment, e.g. core switch

## WHEN TO USE

- Regardless of complexity in installation
- IT environment with sufficient resources

# Simplified setup, but restricted view

## DESCRIPTION

Staging Checkmk instance checks against simulation of productive environment

## HOW IT WORKS

- Create one Checkmk staging instance in parallel to productive system
- Add simulation for SNMP hosts
- Add simulation for agent-based hosts
- Direct checks against simulations

## BENEFITS

- Distributed installations can be tested on one central site
- Avoid additional load in production

## DOWNSIDES

- Considerable configuration effort
- Data renewal for every test
- Restricted view on performance

## WHEN TO USE

- If Gold Standard not possible
- IT environment at the edge of available resources

# 2 dimensions, 2 approaches

## SNMP

Simulation of SNMP data

### HOW IT WORKS

- Configure and simulate your SNMP hosts
  - SNMPsim including e.g. latencies and timeouts**or**
  - Use Checkmk SNMP walks
- Direct check towards simulated SNMP host instead of productive host

## AGENT-BASED

Simulation of agent data

### HOW IT WORKS

- Grab productive agent output
- Synchronize agent output into staging environment
- Direct check towards dump instead of live host

# Agenda

1. CREATING STAGING ENVIRONMENTS
2. BEST PRACTICES FOR TESTING UPDATES





BEST PRACTICES

# Testing updates: 4+5 steps

TESTING

- 1 Features
- 2 Incompatible werks
- 3 Version difference
- 4 MKPs and local structure

UPDATE ROUTINE

- 5 Distributed monitoring
- 6 Downtime scheduling
- 7 Resource requirements
- 8 Ability to roll back
- 9 Backups

# Get familiar with new features

## WHAT

- Understand which additional features come with a new release

## WHY

- Only if configured and applied correctly, new features will bring benefits to your installation, e.g. live status encryption as security feature

## HOW

- Compare new features to current landscape and installation
- Decide which features to use and understand how

We recommend

- Read release notes for major releases

# Create script to automate required actions from incompatible werks

## WHAT

- Installing incompatible werks leads to manual configuration efforts to be assessed precisely before executing updates

## WHY

- Werks could require to e.g. re-discover hosts, which needs to be executed after an update

## HOW

- Check for incompatible werks between your current version and the version to be installed
- Check if these incompatible werks apply to your installation
- Understand the resulting effort for your update

We recommend

- Create a script to automate the required actions

# Testing major updates should be considered mandatory

## WHAT

- Understand the version difference between your current installation and the version you are installing

## WHY

- Skipping major updates not only means lack of new features, but also extends the time needed for the next update

## HOW

- Check your current version next to the logo in your instance
- Compare to the version you want to install and understand if you skip a major release

We recommend

- Install every major release
- Testing major updates should be mandatory

# Keep your local file system as clean as possible

## WHAT

- System updates can affect MKP compatibility

## WHY

- MKPs are stored in your local file system
- Local file system is required to enable local changes, e.g. your own developments or changes to existing code
- Local file system structure will not be changed during updates

## HOW

- Have a look at your local directory: Is this art or trash?
- Clean your local file system as much as possible
- Local structure/MKPs: Explicitly test, where possible
- Problems with an update: Always consider 'local' as cause

We recommend

- Keep your local file system as clean as possible
- Test 'local'/MKPs explicitly

## CONCLUSION

# 5 easy recommendations

- Read release notes for major releases
- Create a script to automate the required actions
- Install every major release
- Testing major updates should be mandatory
- Keep your local file system as clean as possible



Thank you



**tribe29 GmbH**  
Kellerstraße 29  
81667 München  
Deutschland

**Web** — [tribe29.com](https://tribe29.com)  
**E-Mail** — [mail@tribe29.com](mailto:mail@tribe29.com)