FASTER AND EASIER LIFECYCLE HANDLING OF WEBMETHODS PLATFORM

Dave Pemberton
Global Consulting Services
webMethods Practice Manager UK & Nordic

LIFECYCLE HANDLING OF THE PLATFORM
WHAT DO WE MEAN

- Deployments of custom code/configuration
  - IS Packages
  - APIs
  - BPM Processes
  - Tasks, etc.

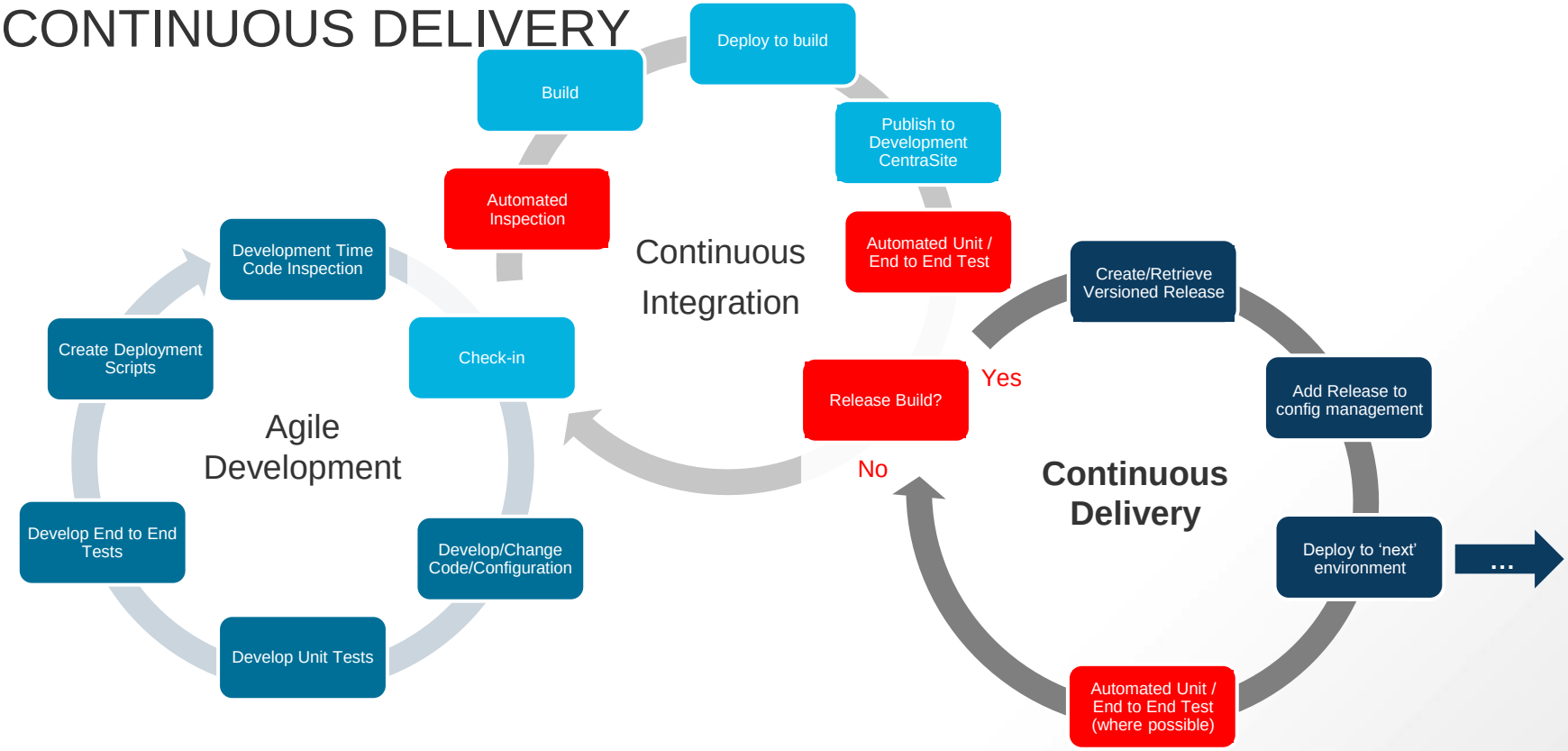
- WebMethods Software
  - Upgrading
  - Patching
  - Containerization

- Zero Downtime?

DEPLOYMENTS OF CUSTOM CODE/CONFIGURATION

CONTINUOUS DELIVERY

DEPLOYMENTS OF CUSTOM CODE/CONFIGURATION
CONTINUOUS DELIVERY



WEBMETHODS SOFTWARE

UPGRADING / PATCHING

WHY UPGRADE…?

Support Lifecycle

New Features

Cloud

DEVOPS
FOUNDATIONS

AGILE

- Deliver Fast
- Deliver Often
- Deliver Right

BUILD AUTOMATION

- Reliability
- Consistency
- Predictability

DEPLOY AUTOMATION

- Reduce Time
- Reduce Errors
- Reduce down time

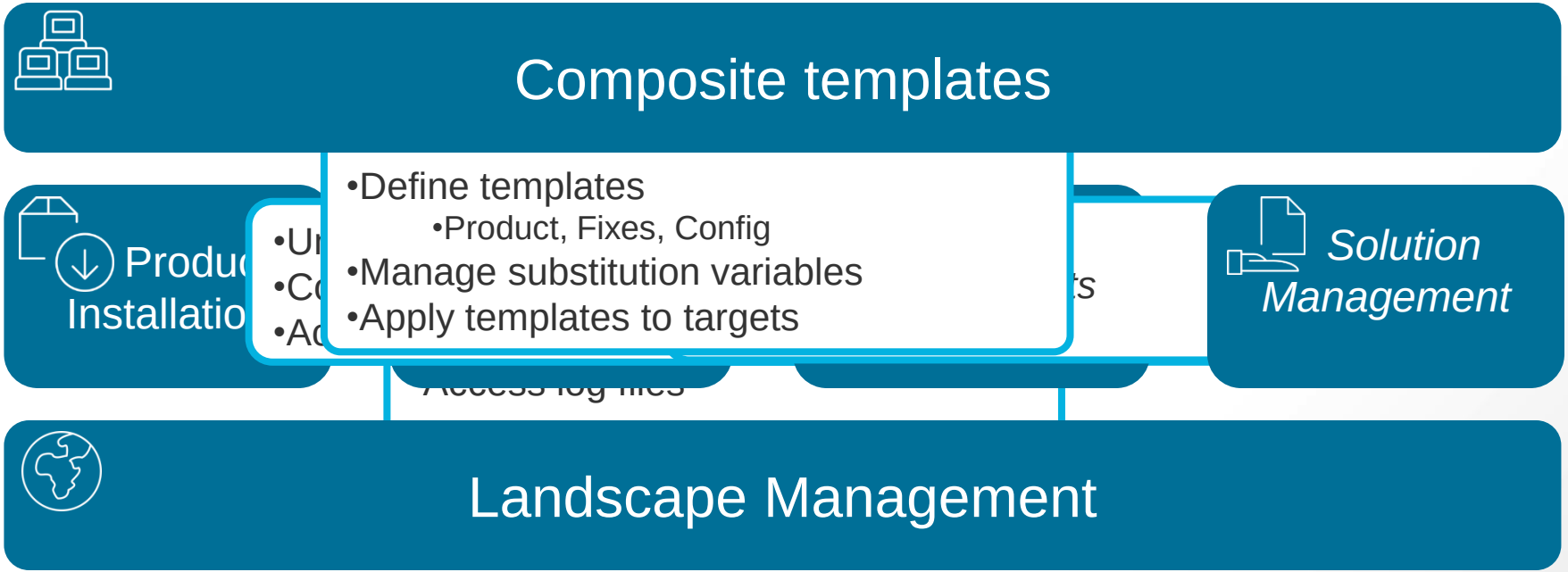
TEST AUTOMATION

- Improve Quality
- Accelerate QA

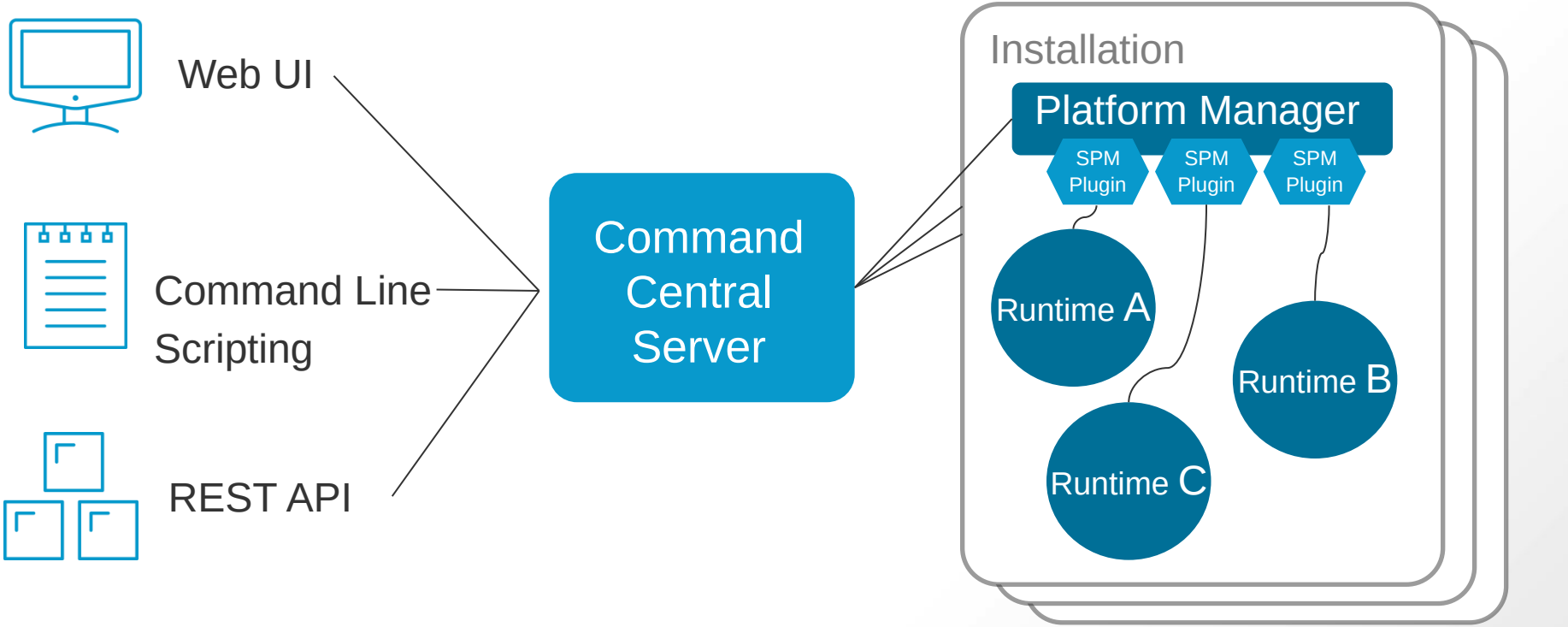
AUTOMATED PROVISON

- Reliability
- Reduce Risks

COMMAND CENTRAL SCOPE



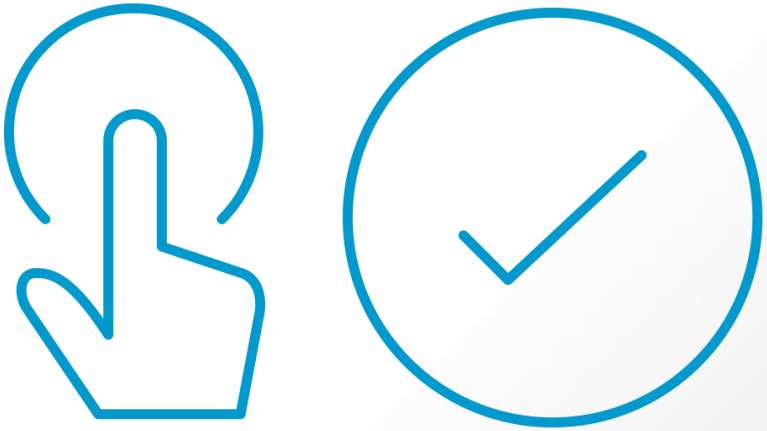
COMMAND CENTRAL ARCHITECTURE (TYPICAL)



CUSTOMER EXAMPLE
BUSINESS DRIVERS, OBJECTIVES

- Challenges and drivers for change
  - Huge complex landscape
  - 300+ instances globally
  - Ad-hoc patching process
  - Frequent provisioning requests
  - Turnaround times
  - Validation and documentation
  - Need for automation (end to end)
  - Central unified management
- Objectives
  - Centrally managed and monitored landscape
  - Automated provisioning, patching and version upgrades

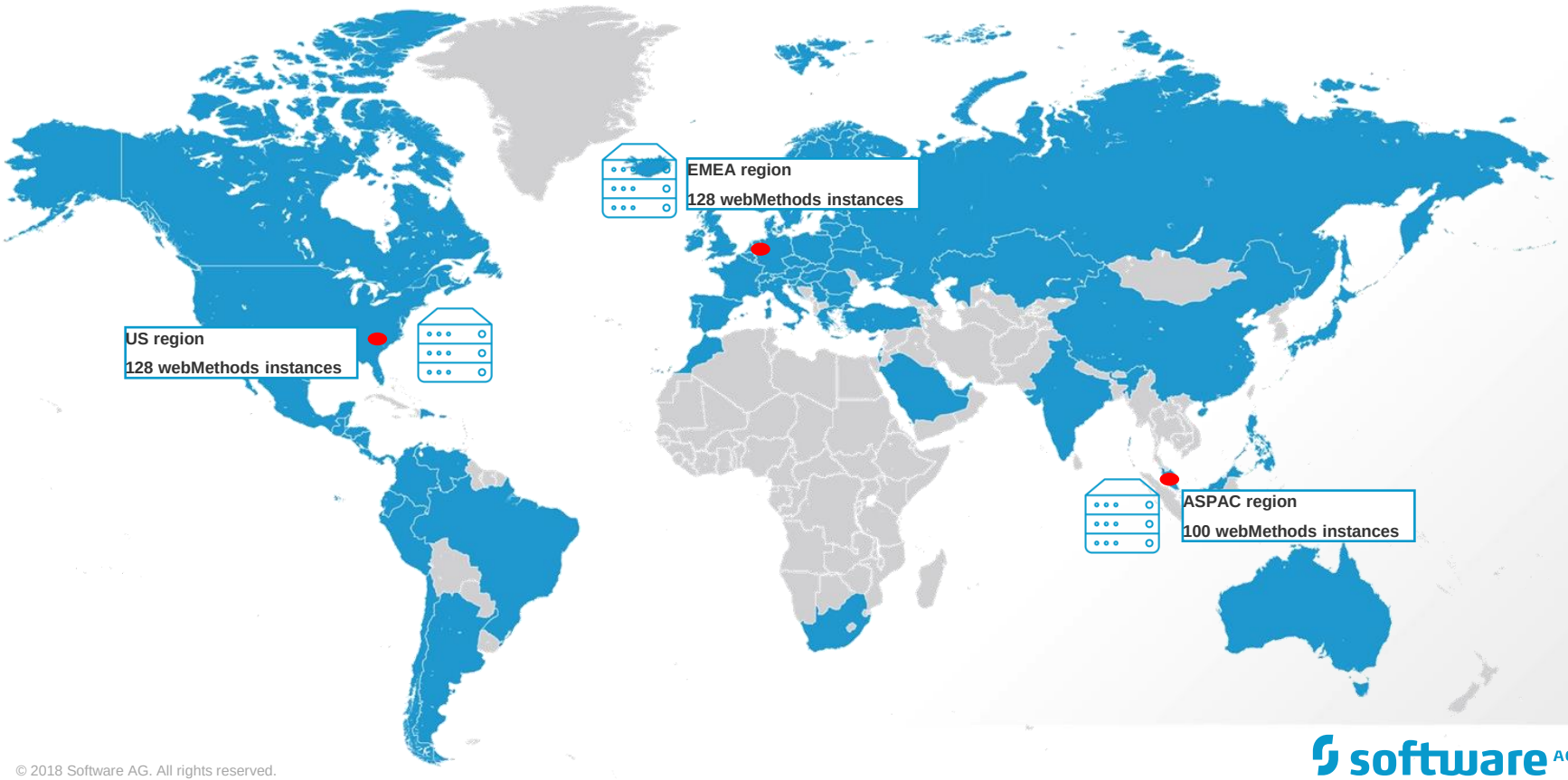
SOLUTION IS



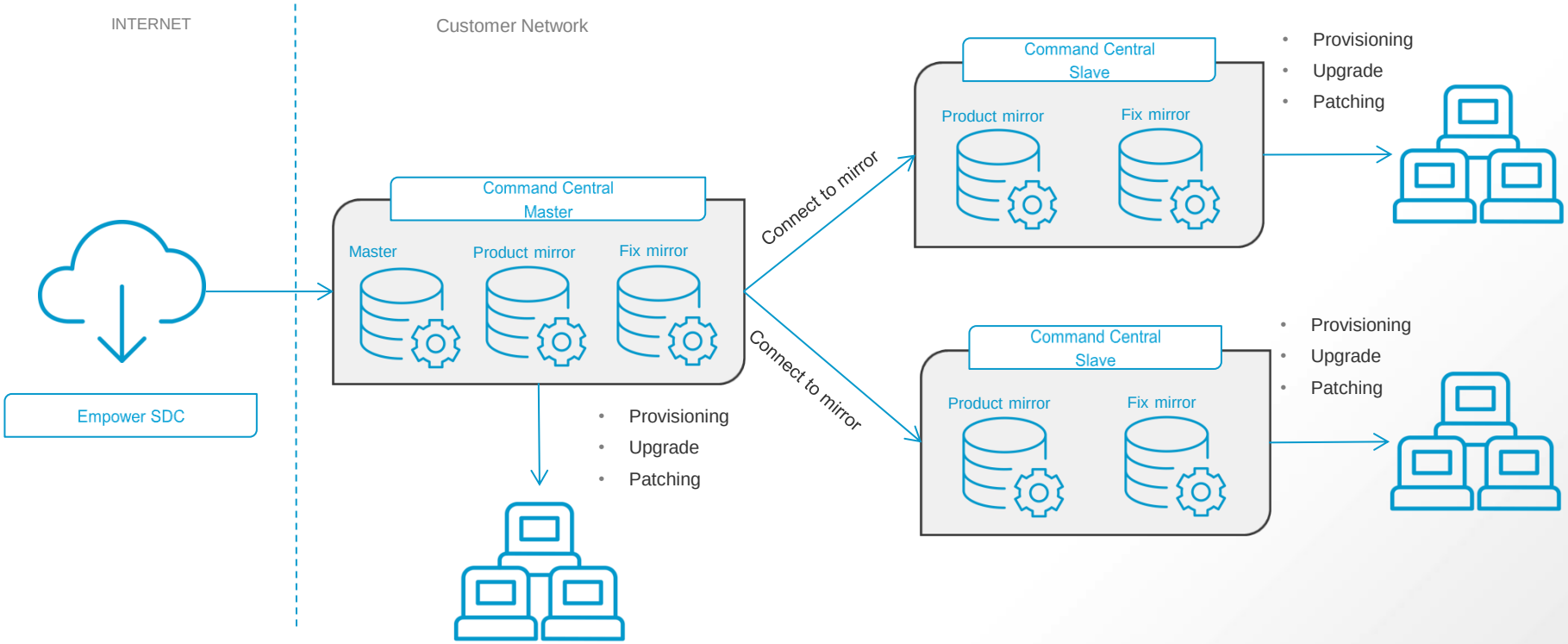
COMMAND CENTRAL

software AG

CUSTOMER GLOBAL LANDSCAPE



CUSTOMER COMMAND CENTRAL SETUP



TEMPLATES
TYPES OF TEMPLATES

Provisioning template

- 400+ configurations
- Single template to handle multiple type environment types, e.g. standalone, cluster etc.
- Pluggable custom solutions invoked via composite templates

83% manual activities to 0%
Effort less & single click
Less risky
Single job for provisioning, patching, configuration and post installation activities

Upgrade template

- 9.7 > 9.12
- 300 instances
- Data migration
- Support for all environment types
- Narrow upgrade window

Upgrade in under 2 hours
Write once and repeat for similar environments
0% manual
Single click or command for entire upgrade flow

Patch template

- 300 instances
- Support for all multiple environments e.g. standalone, cluster etc.
- Pluggable custom solutions invoked via composite templates

What used to take 6 months now takes 2 weeks
Parallelized execution model
Fully automated, flexible, modular templates

software AG

SAG COMMAND CENTRAL TEMPLATES

https://github.com/SoftwareAG/sagdevops-templates

Features Business Explore Marketplace Pricing Search / Sign in or Sign up

SoftwareAG / sagdevops-templates

Watch 19 Star 9 Fork 23

Code Issues 2 Pull requests 1 Projects 0 Wiki Insights

Join GitHub today

GitHub is home to over 28 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

Overview

Sergei Pogrebnyak edited this page on 12 Jun · 4 revisions

Command Central provides tooling for Software AG products provisioning and migration, but it requires a significant learning curve and implementation effort for developing and testing custom templates for specific requirements.

> The goal of this project is to provide default templates library for all supported Software AG products, including comprehensive coverage for all supported configurations.

Pages 11

- Overview
- Importing templates library
- Using default templates
- Building Command Central

Micro Templates for provisioning Software AG products

The following table lists the run-time micro templates for Software AG products.

Template alias	Provisions
sag-abe	Asset Build Environment
sag-apama-correlator	Apama correlator instance
sag-des	Digital Event Services
sag-des-config	Digital Event Services configuration
sag-designer-services	Designer Service Development
sag-designer-cloudstreams	Designer Cloudstreams Development
sag-exx-broker	EntireX Broker
sag-exx-c-rpc-server	EntireX C RPC server
sag-exx-java-rpc-server	EntireX Java RPC server
sag-exx-xml-rpc-server	EntireX XML RPC server
sag-infradc	Infrastructure Data Collector
sag-is-server	webMethods Integration Server instance
sag-is-cluster	webMethods Integration Server stateless cluster
sag-is-statefull-cluster	webMethods Integration Server statefull cluster
sag-is-config	webMethods Integration Server configurations
sag-is-cloudstreams	Cloudstreams on Integration Server or Microservices runtime
sag-is-applatform	Application Platform on Integration Server or Microservices runtime
sag-msc-server	webMethods Microservices Runtime
sag-tc-server	Terracotta BigMemory server
sag-tc-cluster	Terracotta BigMemory cluster
sag-tdb-server	Terracotta DB server
sag-um-server	Universal Messaging server
sag-um-cluster	Universal Messaging cluster
sag-um-config	Universal Messaging configuration

software AG

WEBMETHODS SOFTWARE

CONTAINERISATION

THE CHALLENGE

Multiplicity of Stacks

webMethods
Integration Server + Terracotta + UM

User DB
postgresql + pgv8 + v8

Static website
nginx 1.5 + modsecurity + openssl + bootstrap 2

Queue
Redis + redis-sentinel

Analytics DB
hadoop + hive + thrift + OpenJDK

Background workers
Python 3.0 + celery + pyredis + libcurl + ffmpeg + libopencv + nodejs + phantomjs

Web frontend
Ruby + Rails + sass + Unicorn

API endpoint
Python 2.7 + Flask + pyredis + celery + psycopg + postgresql-client

Do services and apps interact appropriately?

Multiplicity of hardware environments

Development VM

QA server

Customer Data Center

Public Cloud

Disaster recovery

Production Servers

Production Cluster

Contributor's laptop

Can I migrate smoothly and quickly?

software AG

RESULTS IN N X N COMPATIBILITY NIGHTMARE

	Development VM	QA Server	Single Prod Server	Onsite Cluster	Public Cloud	Contributor's laptop	Customer Servers
Static website	?	?	?	?	?	?	?
Web frontend	?	?	?	?	?	?	?
webMethods	?	?	?	?	?	?	?
User DB	?	?	?	?	?	?	?
Analytics DB	?	?	?	?	?	?	?
Queue	?	?	?	?	?	?	?

software AG

DOCKER IS A SHIPPING CONTAINER SYSTEM FOR CODE

Multiplicity of Stacks

Static website | User DB | Web frontend | Queue | webMethods

Do services and apps interact appropriately?

An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container…

…that can be manipulated using standard operations and run consistently on virtually any hardware platform

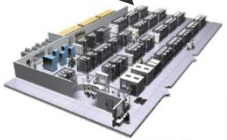
Can I migrate smoothly and quickly

Multiplicity of hardware environments

Development VM

QA server

Customer Data Center

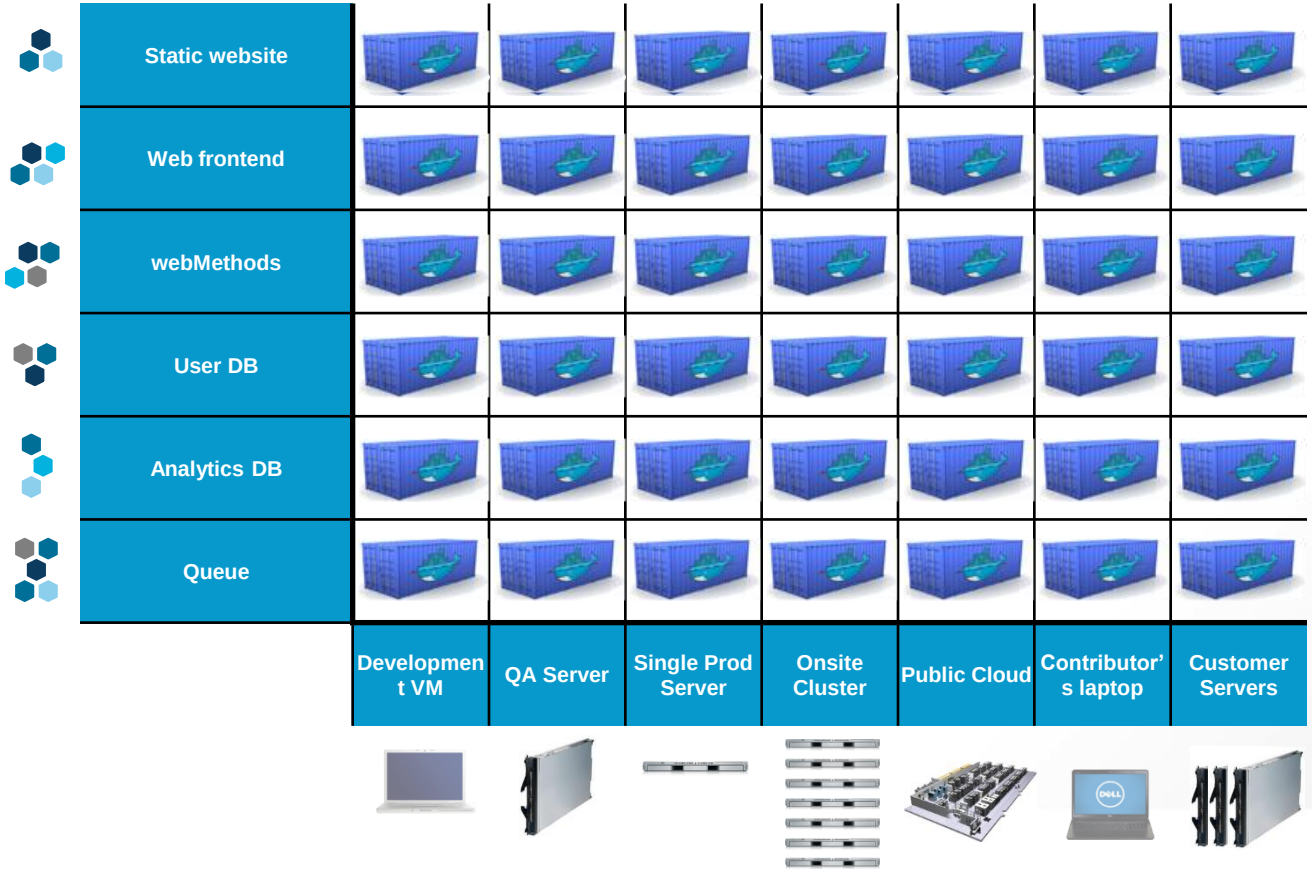


Public Cloud

Production Cluster

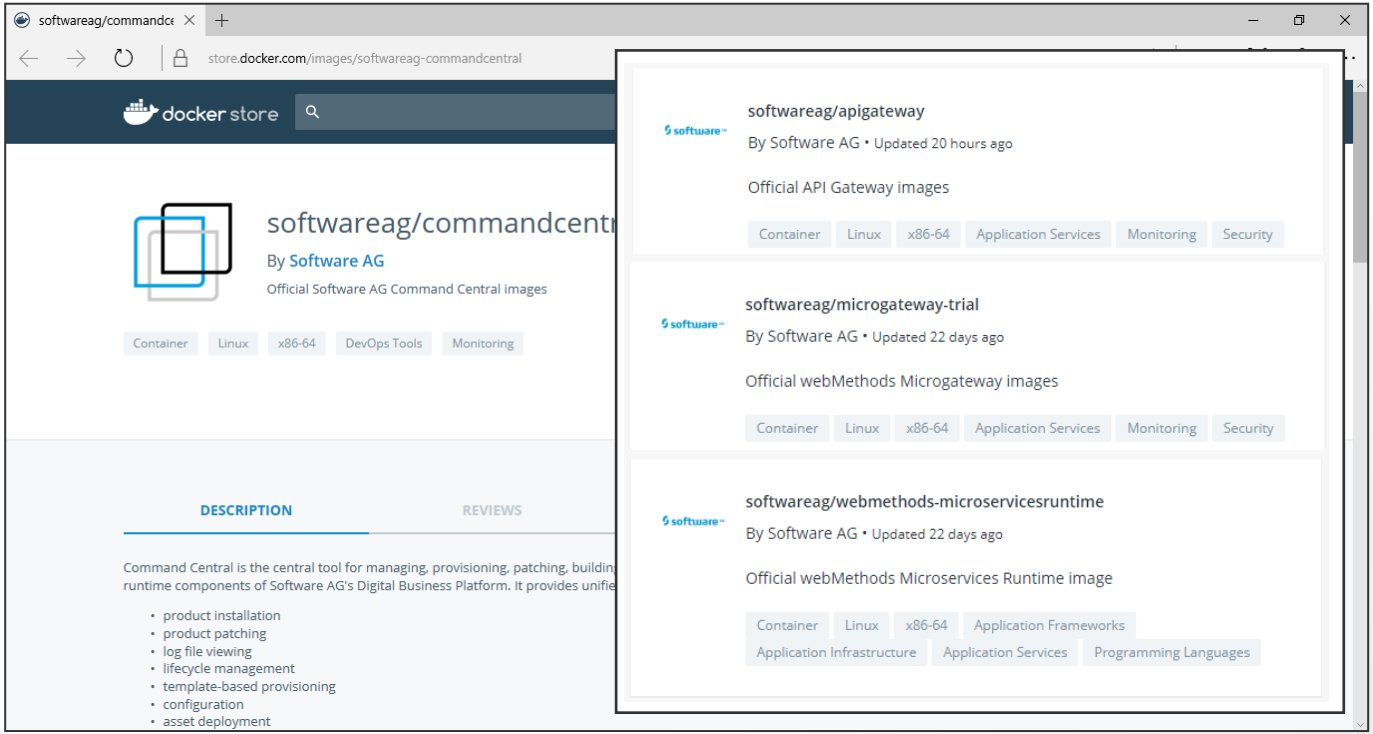
software AG

DOCKER SOLVES THE NXN PROBLEM



software AG

SOFTWARE AS AN APPLIANCE
NO MORE BIG INSTALLS?!



COMMAND CENTRAL – DOCKER BUILDER
ARE THE DAYS OF BIG INSTALLS DOOMED?!

softwareag/commandcentra | GitHub - SoftwareAG/sa

GitHub, Inc. [US] github.com/SoftwareAG/sagdevops-cc-docker-builder

Command Central Docker Builder

This project demonstrates how to build docker images using Command Central Docker Builder.

Overview

Command Central Docker Builder is an tool provided by Software AG on Docker Store. It leverages Command Central templates to run provisioning operations during Docker image build, driven from a standard Dockerfile.

Configuration

Login to Docker Store with your Docker ID, open https://store.docker.com/images/softwareag-commandcentral and accept license agreement to get access to Command Central images.

Login to Docker with your Docker ID from your console and verify you can download the images:

```
docker login
docker pull store/softwareag/commandcentral:10.1.0.1-server
```

Modify init.yaml to configure product and fix repositories, poiting to Master 10.1 repostories on Empower using your Empower credentials or point to your local 10.1 mirrors

```
  product:
    products:
```


software AG

DOCKER CONTAINER
BUILDING YOU OWN USING CC DOCKER BUILDER

dockerfile

```
FROM softwareag/commandcentral-builder:10.3 as builder
RUN $CC_HOME/provision.sh myTemplate && $CC_HOME/cleanup.sh
FROM centos:7 as base
COPY -from=builder /opt/softwareag /opt/softwareag
EXPOSE 1234
ENTRYPOINT /opt/softwareag/some/bin/runme.sh
```

Template.yaml

```
FROM softwareag/commandcentral-builder:10.3 as builder
RUN $CC_HOME/provision.sh myTemplate && $CC_HOME/cleanup.sh
FROM centos:7 as base
COPY -alias: myTemplate
templates:
  myContainer:
     fixes: ALL             # install all fixes
     products:
        myProduct:          # install myProduct
           myInstance:      # create myInstance of myProduct
              my.port: 1234                   # port configuration
              configuration:
                configurationType:
                  configurationInstance:
                    property1: value1         # configuration property1
                    property2: value2         # configuration property2
layers:
   runtimes:
      productRepo: ${repo.product}            # product repository
      fixRepo:     ${repo.fix}                # fix repository
      templates:   myContainer
provision:
   default:
      runtime: ${nodes}from=builder /opt/softwareag /opt/softwareag
EXPOSE 1234
ENTRYPOINT /opt/softwareag/some/bin/runme.sh
```


DOCKER CONTAINER
BUILDING YOU OWN – KEY POINTS

- The template should **define only a single runtime instance**. Different product runtimes should run as different Docker containers and thus should be built as separate Docker images
- Baking as much configuration as possible into the Docker image allows to achieve (almost) **immutable infrastructure**, which brings a lot of benefits for running containerized applications. Mutable configuration aspects can be addressed by leveraging product specific capabilities like dynamic re-configuration using environment variables.
- The templates used for building Docker images are the same templates that can be used for traditional provisioning. This allows developing and testing templates iteratively outside of the Docker build process and then use them for both traditional and containerized deployments.
-

DOCKER CONTAINERS
THE BUILD PROCESS

- The builder process is as simple as running **docker build** command in the folder that contains the Dockerfile:

```
docker build -t myimage
```


DOCKER CONTAINERS
JUST INTEGRATION SERVER?

is_container Script Commands

The following table provides descriptions of for the commands that an be used with the is_container.sh/bat script.

Command	Description
`createDockerfile`	Creates a Dockerfile for a base Integration Server instance, including "Default" and "Wm" packages only.
`createLeanDockerfile`	Creates a Dockerfile for a base Integration Server instance, including the "Default" package and the "Wm" packages that are required for core Integration Server such as WmRoot, WmPublic, and WmCloud.
`createPackageDockerfile`	Creates a Dockerfile for custom packages.
`build`	Executes Docker build using the provided Dockerfile to build a base image of Integration Server
`buildPackage`	Executes Docker build using the provided Dockerfile to build an image of Integration Server that contains custom packages.
`saveImage`	Saves the image from the local Docker registry into a tar file specified by the file.path parameter.
`loadImage`	Loads the image specified in the file.path parameter into a local Docker registry.
`pushImage`	Pushes Integration Server image created for the on-premise Integration Server into the webMethods Cloud or Docker registry.
`help`	Displays usage information for each command.

Environment Variable	Description
`EXTERNALIZE_PACKAGES`	When set to true, instructs the Integration Server running in the Docker container to load the packages located in one of the following directories at startup: ■ *<HOST_DIR>/<SERVICE_NAME>*/packages ■ *<HOST_DIR>/<INSTANCE_NAME>*/packages Where *HOST_DIR SERVICE_NAME*, and *INSTANCE_NAME* are the values set for the ENV variables of the same name. If **`SERVICE_NAME`** is supplied,Integration Server looks in *<HOST_DIR>/<SERVICE_NAME>*/packages. Only if **`SERVICE_NAME`** is not supplied, does Integration Server look in *<HOST_DIR>/<INSTANCE_NAME>*/packages. The default value of **`EXTERNALIZE_PACKAGES`** is false. The content of the packages directory must be a folder where the folder name is the package name. The contents of the *packageName* folder must match the structure of Integration Server packages. That is the packages located in the packages directory cannot be an archive file, such as *.zip or *.7z. For information about the structure of an Integration Server package, see How the Server Stores Package Information.
`HOST_DIR`	The path to the mounted directory on the HOST machine to which to write the files. Passing the **`HOST_DIR`** parameter and value or setting it as an ENV variable with the **`docker run`** command externalizes the logs and configuration artifacts of Integration Server.
`PERSIST_LOGS`	If set to true, the Integration Server running inside the Docker container persists the log files to *HOST_DIR/SERVICE_NAME*/logs where *SERVICE_NAME* is

DOCKER CONTAINERS
SUMMARY

- You can use Command Central DevOps tooling for traditional on-premises provisioning as well as containerized deployments.
- You can build managed (by Command Central) and unmanaged Docker images for 10.x releases, having great control of what exactly goes into your images to satisfy your security requirements or requirements from your orchestration engine.

For up-to-date detailed instructions visit:

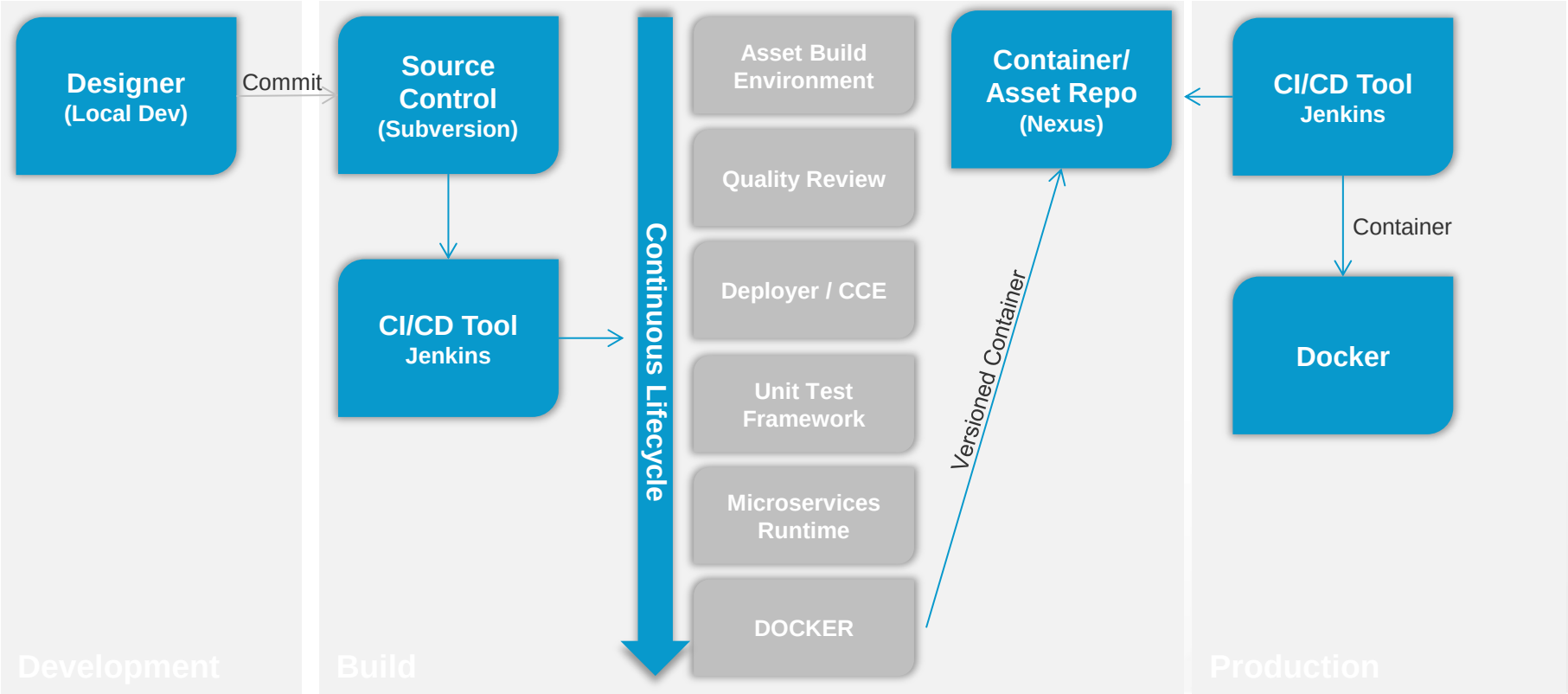
https://github.com/SoftwareAG/sagdevops-templates

software AG

MIXING IT UP

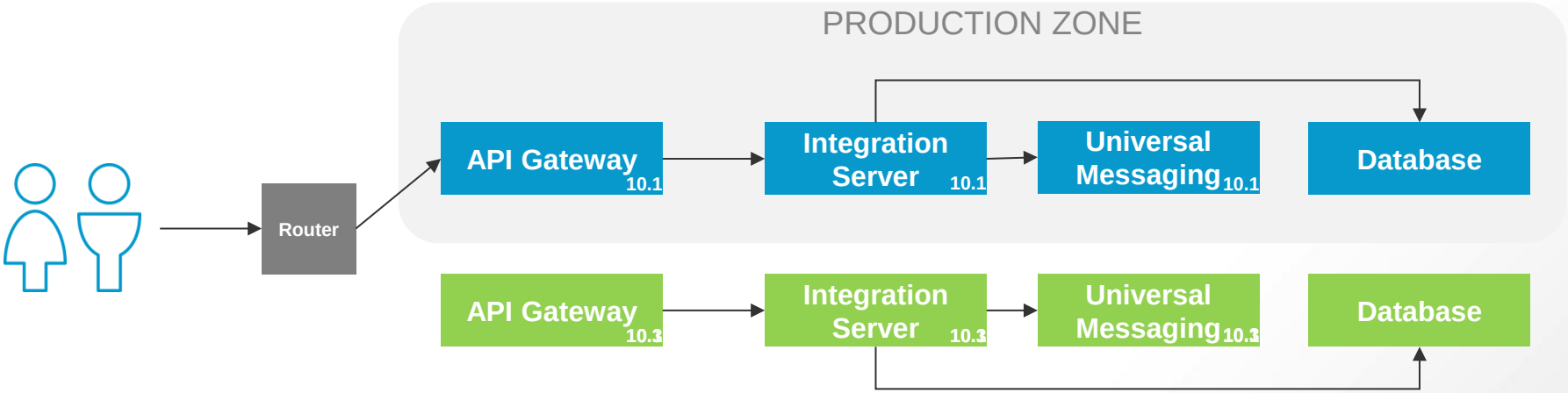
Devops + Containers!

MIXING IT UP
BUILDING / DEPLOYING CONTAINERS!



ZERO DOWNTIME?

ZERO DOWNTIME DEPLOYMENT & RELEASE
BLUE-GREEN DEPLOYMENT - UPGRADE



CUSTOMER EXAMPLE
AEMO

The Australian Energy Market Operator
Responsible for Australia's largest gas and electricity markets and power systems

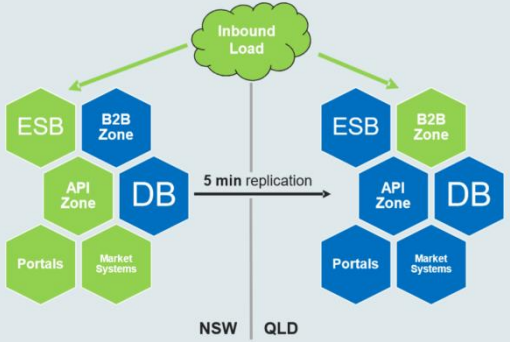
Discipline 3: Mature High Availability Practices

Site Transfer
Validate **near-instant** recovery every 6 weeks

Automation
Minimise manual deployment and transfer steps

Hot Standby
Applications in the inactive site are always **running**

The AEMO Site Transfer



Transfer by Bays