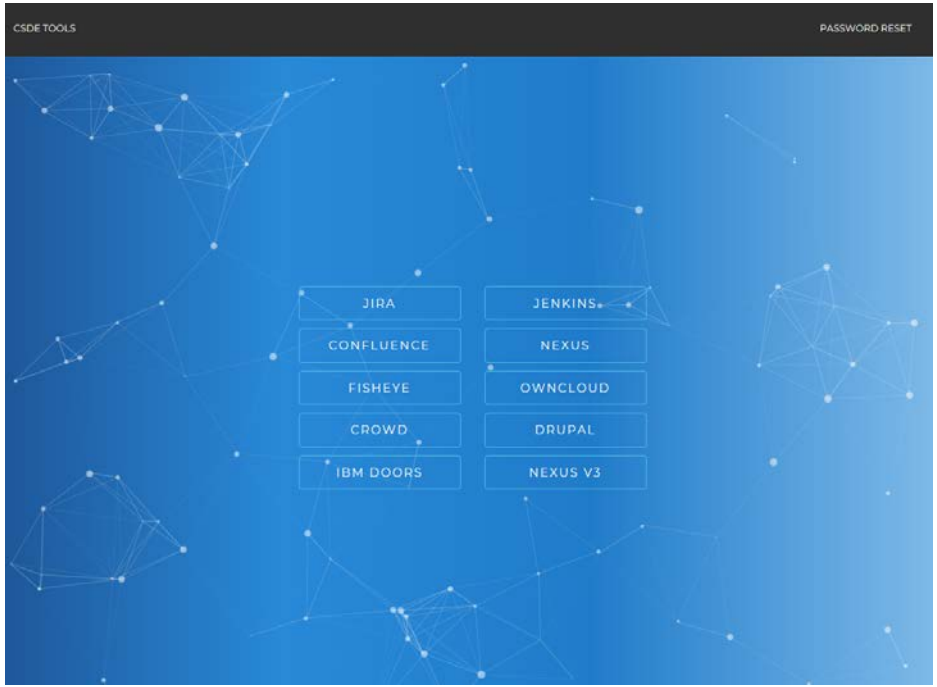
Mastering major CSDE upgrade

Abstract



https://csde.esa.int

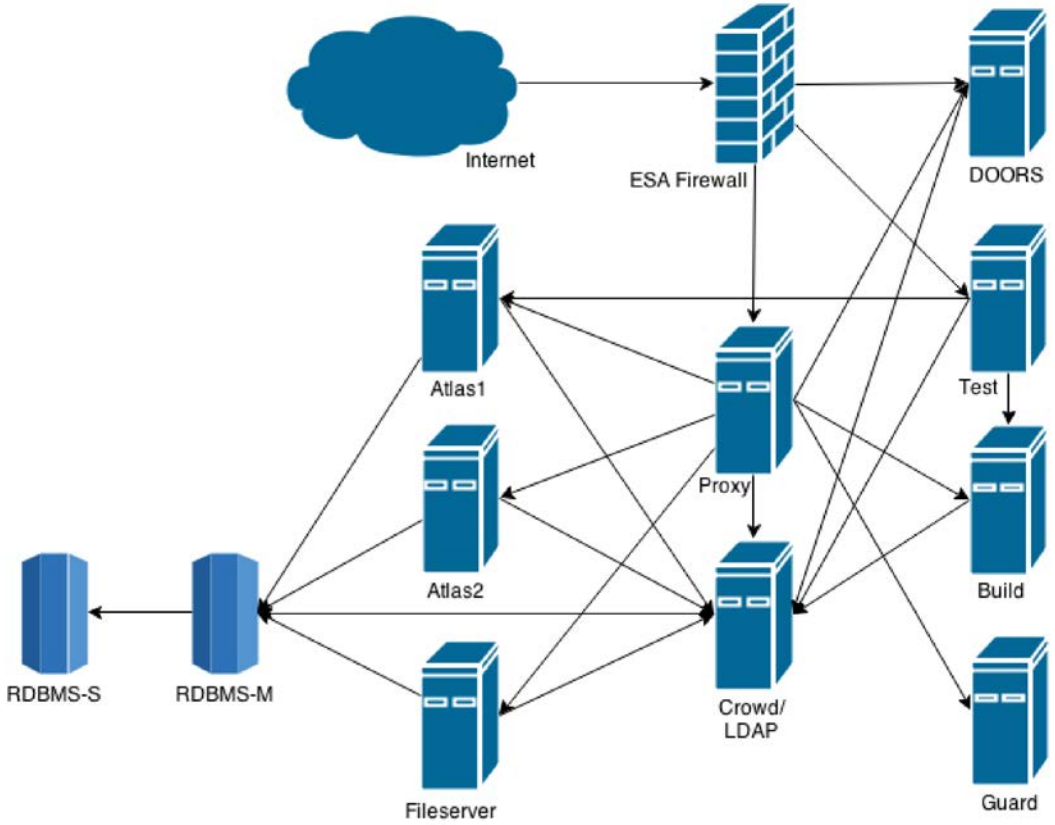
Objective:

The main task of the project was to upgrade the primary applications at CSDE.

Background and justification:

After the our IT audit, we came to the conclusion that the internal
IT infrastructure must be re-engineered to make it flexible, scalable and applicable on any platform.

Brief description of the former system



Brief description of the former system

Unfortunately, the current state of the system has ceased to correspond to the level of tasks being solved.

We were involved in process of system upgrade since it was decided to modernize the system.

Achievments and status:

We modified the IT infrastructure and introduced an innovative DevOps approach giving us the opportunity to further improvement and adding new functionality without downtime and without affecting the user activities:

- all applications are launched in Docker containers
- an Ansible - continuous management(CM) system was deployed and used to deal with infrastructure, hosts and applications

Achievments and status:

- the monitoring system is completely rebuilt and new subsystems are integrated into it: Zabbix and Graylog - for centralized storing and analyzing logs
- as the reverse proxy we've used nginx for ssl termination and all modern security technologies were used
- dozens of Ansible playbooks and Bash/Python scripts with a total volume exceeding some thousands lines of code have been written and debugged

1

APPLICATIONS LIST

Application list

The work consisted of two main parts:

- changes in the development environment
- changes in the production environment

The summary listing of applications could be found below:

- Atlassian Jira (issue tracking)
- Atlassian Confluence (enterprise wiki)
- Atlassian Fisheye/Crucible source code management/review)
- Atlassian Crowd (single sign-on)
- Sonatype Nexus (repository management for artifacts)
- Sonatype Nexus3 (repository management for artifacts)
- Drupal (content management system)
- ownCloud (data exchange)
- Jenkins (automated build/testing and continuous integration)
- Forgerock OpenDJ (service server for LDAP support)

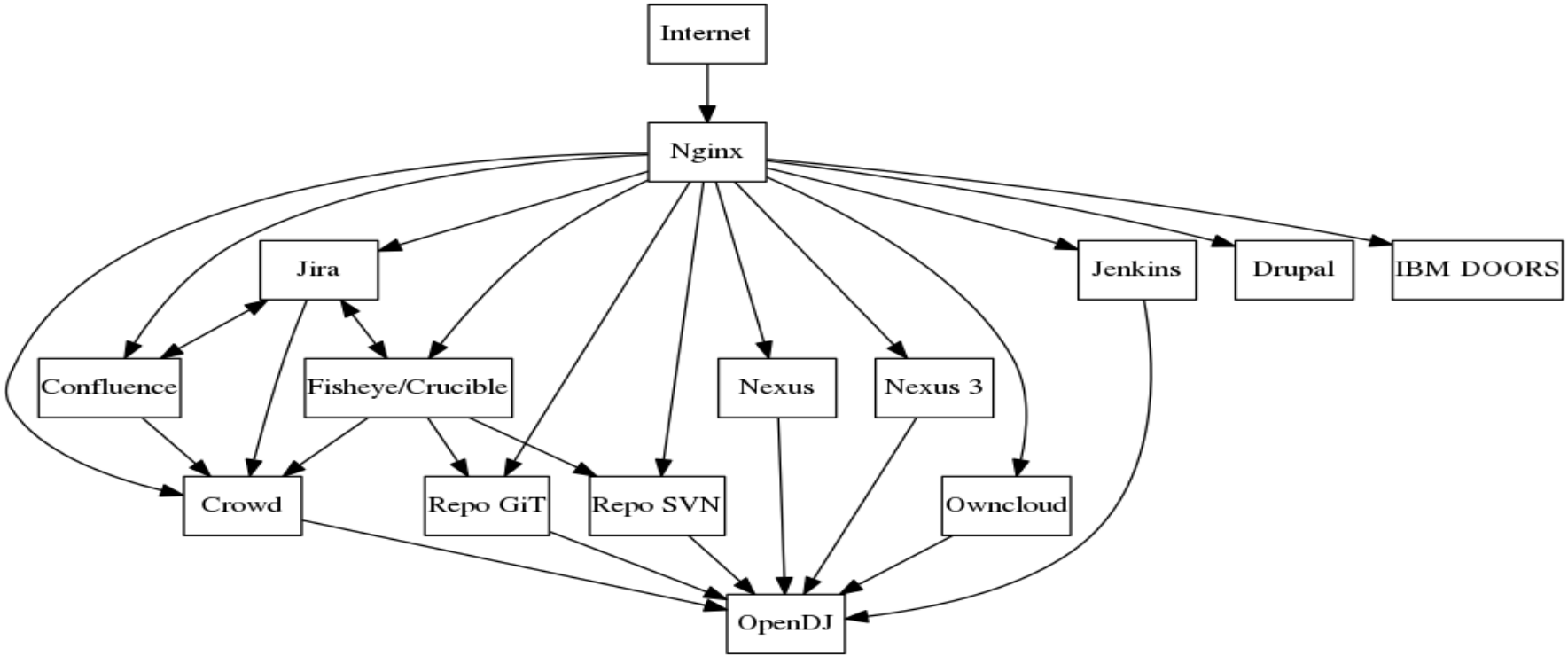
Application list

- IBM Rational DOORS (requirements management)
- GIT (version control)
- SVN (version control)

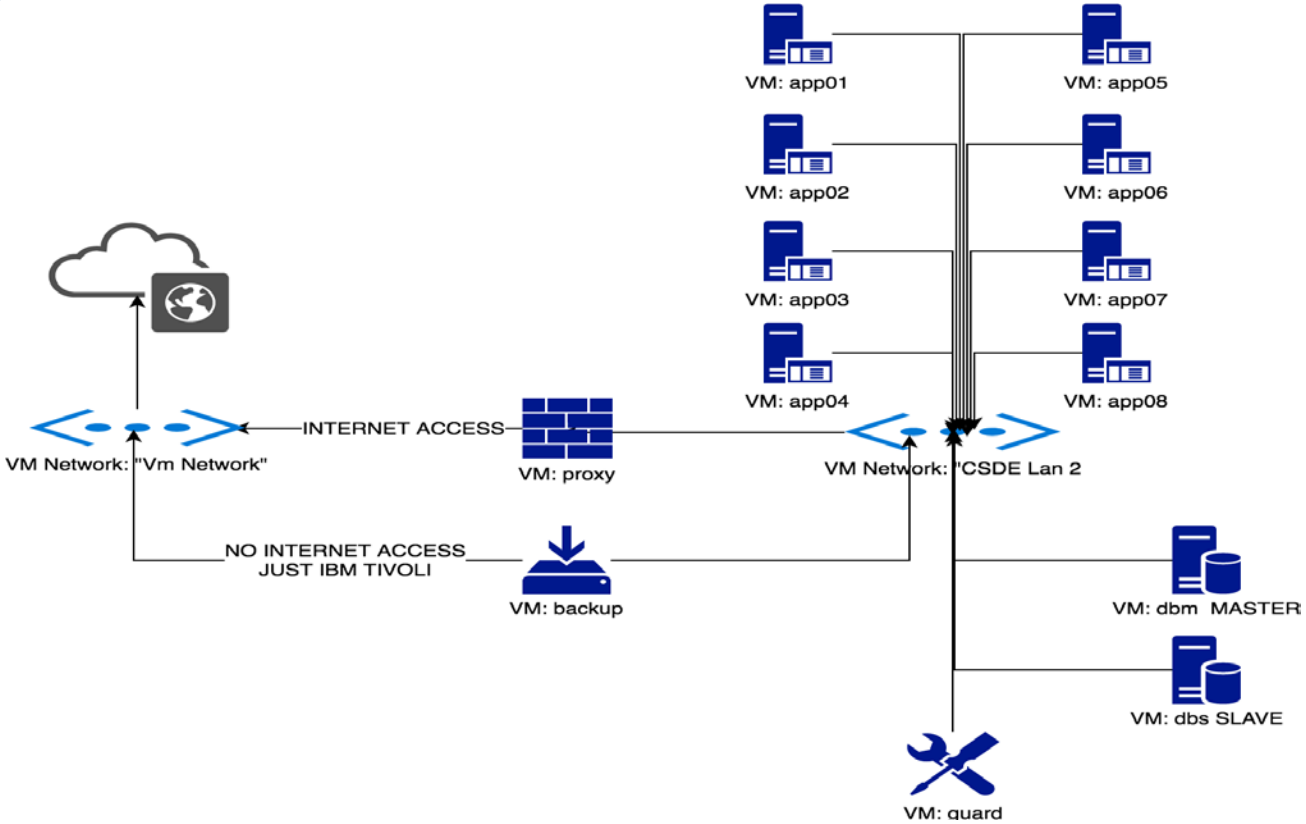
Below you could find also some applications that were upgraded/installed:

- Postgresql master/slave (SQL database)
- Docker (container management)
- Docker Registry (local Docker repository)
- Zabbix (monitoring system)
- Graylog (logs collecting/managing)
- Ansible (continuous management)
- Nginx Proxy (reverse proxy)
- Backup

Application scheme



Infrastructure scheme

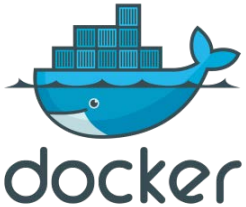


2

MAJOR TECHNOLOGIES

Major Technologies

Docker

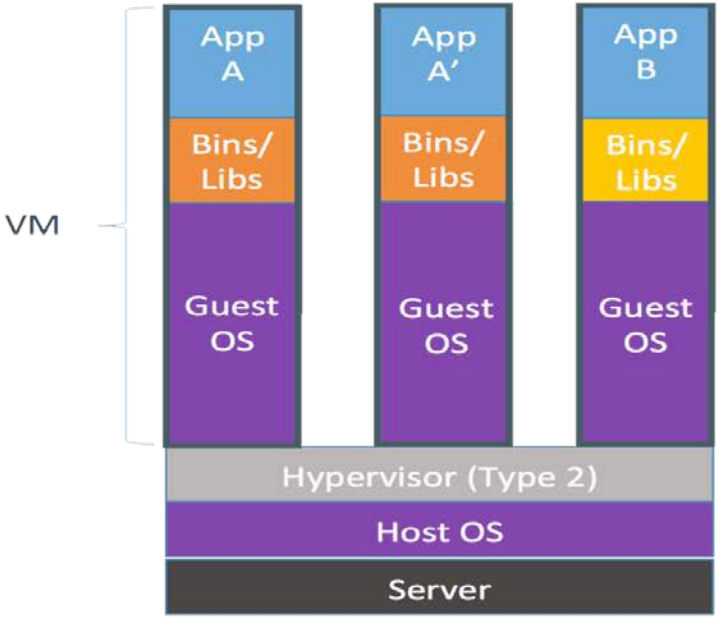


Zabbix

Ansible

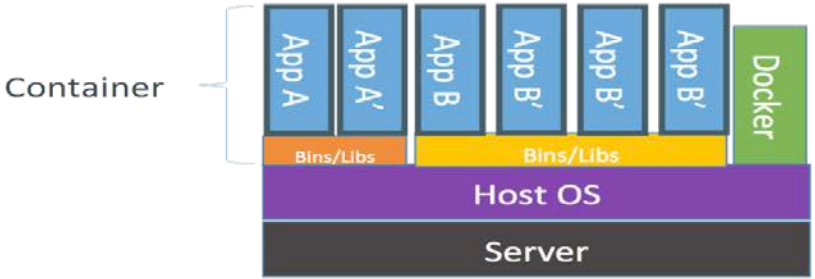
Graylog

Brief info



Containers are isolated, but share OS and, where appropriate, bins/libraries

...result is significantly faster deployment, much less overhead, easier migration, faster restart



Brief info

Docker really makes it easier to create, deploy, and run applications by using containers. And containers allow us to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package. By doing so, we can be assured that the application will run on any other Linux machine regardless of any customized settings that machine might have that could differ from the machine used for writing and testing the code. I.e. all we need in to run application at any host with arbitrary OS – just running Docker Container Engine(daemon).

If we need to perform an upgrade we can easily make the necessary changes to Docker containers, test them, and implement the same changes to our existing containers. This sort of flexibility is a key advantage of using Docker.

Brief info

Docker makes sure each container has its own resources that are isolated from other containers. We have various containers for separate applications running completely different stacks. Aside from this, effectively removing applications from server is quite difficult and may cause conflicts with dependencies. However, Docker helps us ensure clean app removal since each application runs on its own container. If we no longer need an application, we can simply delete its container.

On top of these benefits, Docker also ensures that each application only uses resources (CPU, memory and disk space) that have been assigned to them. A particular application won't hog all of available resources, which would normally lead to performance degradation or complete downtime for other applications

Containers list

Atlassian Confluence	Sonatype Nexus
Atlassian Crowd	Sonatype Nexus 3
Atlassian Fisheye/Crucible	OpenDJ
Atlassian Jira	ownCloud
Jenkins	Git
Drupal	SVN

Dockerfile for Sonatype Nexus 3:


```
FROM ubuntu:xenial
MAINTAINER IT Consulting Group B.V. <info@itcgr.com>

#JAVA ENV
ENV VERSION  < HIDDEN >
ENV UPDATE  < HIDDEN >
ENV BUILD < HIDDEN >
ENV SIG < HIDDEN >
ENV JAVA_HOME /usr/lib/jvm/java-${VERSION}-oracle
ENV JRE_HOME ${JAVA_HOME}/jre
ENV OPENSSL_VERSION 1.0.2l

#NEXUS ENV
ENV NEXUS_VERSION 3.6.2-01
ENV SONATYPE_DIR /opt/sonatype
ENV NEXUS_HOME ${SONATYPE_DIR}/nexus
ENV NEXUS_DATA /nexus-data
ENV NEXUS_CONTEXT 'nexus3'
ENV SONATYPE_WORK ${SONATYPE_DIR}/sonatype-work
ENV RUN_USER            daemon
ENV RUN_GROUP           daemon
```

Dockerfile for Sonatype Nexus 3:

```
#FOR DEV
RUN touch /etc/apt/apt.conf.d/99fixbadproxy \
   && echo "Acquire::http::Pipeline-Depth 0;" >> /etc/apt/apt.conf.d/99fixbadproxy \
   && echo "Acquire::http::No-Cache true;" >> /etc/apt/apt.conf.d/99fixbadproxy \
   && echo "Acquire::BrokenProxy true;" >> /etc/apt/apt.conf.d/99fixbadproxy

RUN set -x \
    && apt-get update --quiet -o Acquire::CompressionTypes::Order::=gz \
    && apt-get install --quiet --yes --no-install-recommends ca-certificates curl wget\
       gcc libc6-dev libssl-dev make \
    && apt-get clean

#INSTALL JAVA
RUN             curl --silent --location --retry 3 --cacert /etc/ssl/certs/GeoTrust_Global_CA.pem \
                --header "Cookie: oraclelicense=accept-securebackup-cookie;" \
                http://download.oracle.com/otn-pub/java/jdk/"${VERSION}"u"${UPDATE}"-
b"${BUILD}"/"${SIG}"/jre-"${VERSION}"u"${UPDATE}"-linux-x64.tar.gz \
                | tar xz -C /tmp && \
                mkdir -p /usr/lib/jvm && mv /tmp/jre1.${VERSION}.0_${UPDATE} "${JAVA_HOME}"
```

Dockerfile for Sonatype Nexus 3:

```
RUN apt-get remove --purge --auto-remove -y \
        gcc \
        libc6-dev \
        libssl-dev \
        make && \
    apt-get autoclean && apt-get --purge -y autoremove && \
    rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

RUN update-alternatives --install "/usr/bin/java" "java" "${JAVA_HOME}/bin/java" 1 && \
                update-alternatives --install "/usr/bin/javaws" "javaws"
"${JAVA_HOME}/bin/javaws" 1 && \
                update-alternatives --set java "${JAVA_HOME}/bin/java" && \
                update-alternatives --set javaws "${JAVA_HOME}/bin/javaws"

#INSTALL NEXUS
#RUN apk update && apk add openssl && rm -fr /var/cache/apk/*
RUN mkdir -p /opt/sonatype/ \
  && wget https://download.sonatype.com/nexus/3/nexus-${NEXUS_VERSION}-unix.tar.gz
-O - \
  | tar zx -C "${SONATYPE_DIR}" && rm -fr ${SONATYPE_WORK} \
  && mv "${SONATYPE_DIR}/nexus-${NEXUS_VERSION}" "${NEXUS_HOME}"
```

Dockerfile for Sonatype Nexus 3:

```
# configure nexus
RUN sed \
    -e '/^nexus-context/ s:$:${NEXUS_CONTEXT}:' \
    -i ${NEXUS_HOME}/etc/nexus-default.properties

RUN mkdir -p "${NEXUS_DATA}/etc" "${NEXUS_DATA}/log" "${NEXUS_DATA}/tmp"
"${SONATYPE_WORK}"
RUN ln -s ${NEXUS_DATA} ${SONATYPE_WORK}/nexus3
## prevent warning: /opt/sonatype/nexus/etc/org.apache.karaf.command.acl.config.cfg (Permission
denied)
RUN chown -R daemon "${NEXUS_HOME}/etc/"

ENV TINI_VERSION 0.14.0
ENV TINI_SHA  < HIDDEN >

# Use tini as subreaper in Docker container to adopt zombie processes
RUN curl -fsSL https://github.com/krallin/tini/releases/download/v${TINI_VERSION}/tini-static-amd64 -o
/bin/tini && chmod +x /bin/tini \
    && echo "$TINI_SHA  /bin/tini" | sha256sum -c -

COPY entrypoint.sh /
RUN chmod +x entrypoint.sh && chown daemon:daemon entrypoint.sh
VOLUME ${NEXUS_DATA}
```

Dockerfile for Sonatype Nexus 3:

```
EXPOSE  < HIDDEN >
EXPOSE  < HIDDEN >

WORKDIR ${NEXUS_HOME}

ENV JAVA_MAX_MEM 4000m
ENV JAVA_MIN_MEM 1200m
ENV JAVA_MAX_HEAP=768m
ENV EXTRA_JAVA_OPTS ""

ENTRYPOINT ["/bin/tini", "--", "/entrypoint.sh"]
```

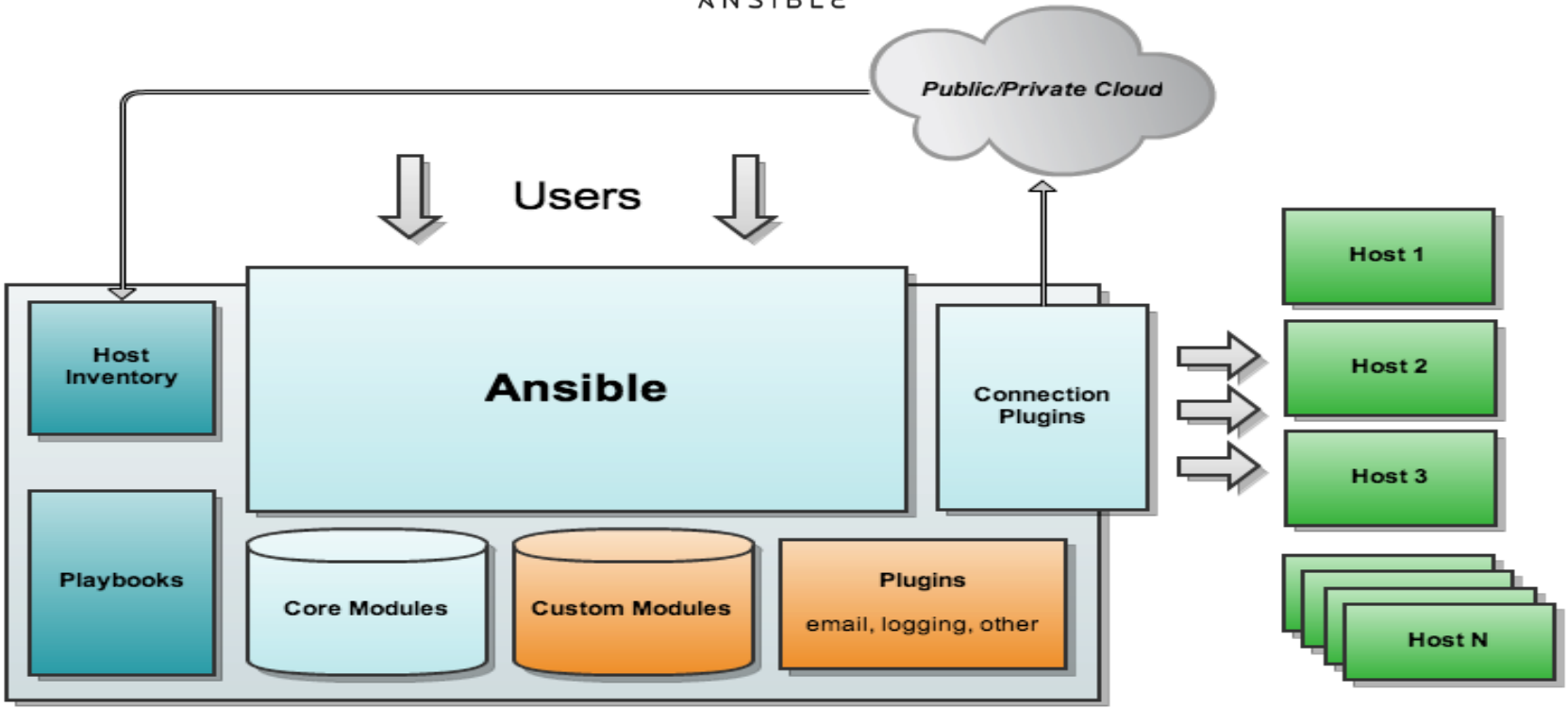
```
#!/bin/sh –e          # Body of entrypoint.sh

[ -d "${NEXUS_DATA}" ] || mkdir -p "${NEXUS_DATA}"
[ $(stat -c '%U' "${NEXUS_DATA}") != 'daemon' ] && chown -R daemon "${NEXUS_DATA}"

# clear tmp and cache for upgrade
rm -fr "${NEXUS_DATA}"/tmp/ "${NEXUS_DATA}"/cache/

[ $# -eq 0 ] && \
    exec su -s /bin/sh -c '/opt/sonatype/nexus/bin/nexus run' daemon || \
exec "$@"
```

Brief info


Public/Private Cloud
Users
Host Inventory
Ansible
Connection Plugins
Playbooks
Core Modules
Custom Modules
Plugins
email, logging, other
Host 1
Host 2
Host 3
Host N

Brief info

We now use Ansible for any task or project that requires repeatable processes and a consistent environment, such as provisioning IT and server infrastructure, installation and configuration of applications, and application deployment.

Ansible uses no agents and no additional custom security infrastructure. Essentially, the desired state of the entire IT infrastructure can be described in the form of Ansible Playbooks, which are files written in a YAML language.

By default, Ansible represents what nodes it manages using a plain text file which organizes the managed machines in custom groups. Rather than just managing one node or system at a time, Ansible will configure each node to the described state.

Brief info

Ansible works by connecting to the nodes over SSH and pushing out small Ansible modules to them. Ansible then executes and removes them when finished. Because these modules are simple Python scripts, and Ansible is agent-less, the target hosts only require an SSH connection and Python installed.

Besides simplicity, which is another advantage of using Ansible vs Chef or Puppet, there are no servers, daemons, or databases required - all that is needed is to install Ansible on the host, Python on the targets, and have SSH access.

Playbooks list

add_users_goups.yml
app_confluence.yml
app_crowd.yml
app_drupal.yml
app_fisheye.yml
app_git.yml
app_jenkins.yml
app_jira.yml
app_nexus.yml
app_nexus3.yml
app_opendj.yml
app_ownclowd.yml
app_svn.yml
backup_config.yml
backup_rsync_all.yml
disable_ipv6.yml
secure_server_graylog.yml
secure_server_ssh.yml
secure_server_ufw.ssh
set_hosts.yml
set_timezone_locale_all.yml
soft_all.yml
soft_app.yml
soft_db.yml
soft_zabbix.yml

Playbook app_nexus3.yml

Below you can find examples of one of the playbooks:

```
# [Author] IT Consulting Group B.V.
# E-mail: info@itcgr.com

- hosts: app05
  user: root
  become: no
  #when: ansible_distribution == 'Ubuntu'
  vars:
    ip_proxy:  < HIDDEN >

  tasks:
    - name: NEXUS | Upload docker directory to the docker host
      synchronize:
        src: /data/service/nexus
        dest: /root
        rsync_opts:
          - "--no-motd"
          - "--exclude=.git”
```

Playbook app_nexus3.yml

```
- name: NEXUS | Remove old image
    docker_image:
      name: csde/nexus
      state: absent
      force: True

  - name: NEXUS | Build new image
    docker_image:
      path: /root/nexus
      name: csde/nexus
      pull: True
```

Playbook app_nexus3.yml

```
- name: NEXUS | Start container
    docker_container:
      name: nexus
      recreate: True
      state: started
      image: csde/nexus
      auto_remove: False
      detach: True
      exposed_ports:
        - < HIDDEN >
      hostname: nexus
      ignore_image: False
      network_mode: host
      published_ports:
        - < HIDDEN >
      restart_policy: on-failure
      restart_retries: 3
      volumes:
        - /data/sonatype-work:/opt/sonatype/sonatype-work
```

Brief info

ZABBIX

Zabbix is an open source enterprise solution, which can fulfill a complex monitoring of an infrastructure (servers, network devices and virtual machines), represent received information in a form of diagrams, watch a device load and performance using own agents (which are supported by almost all operating systems). It has a possibility to send notifications about incidents via an email or an SMS, and can work both through http and https. Zabbix is oriented on an entire infrastructure.

- Zabbix monitors all main protocols (HTTP, FTP, SSH, POP3, SMTP, SNMP, MySQL, etc)
- Alerts in e-mail and/or SMS
- Very good web interface
- Native agent available on Windows, OS X, Linux, FreeBSD, etc
- Multi-step web application monitoring (content, latency, speed)

Brief info

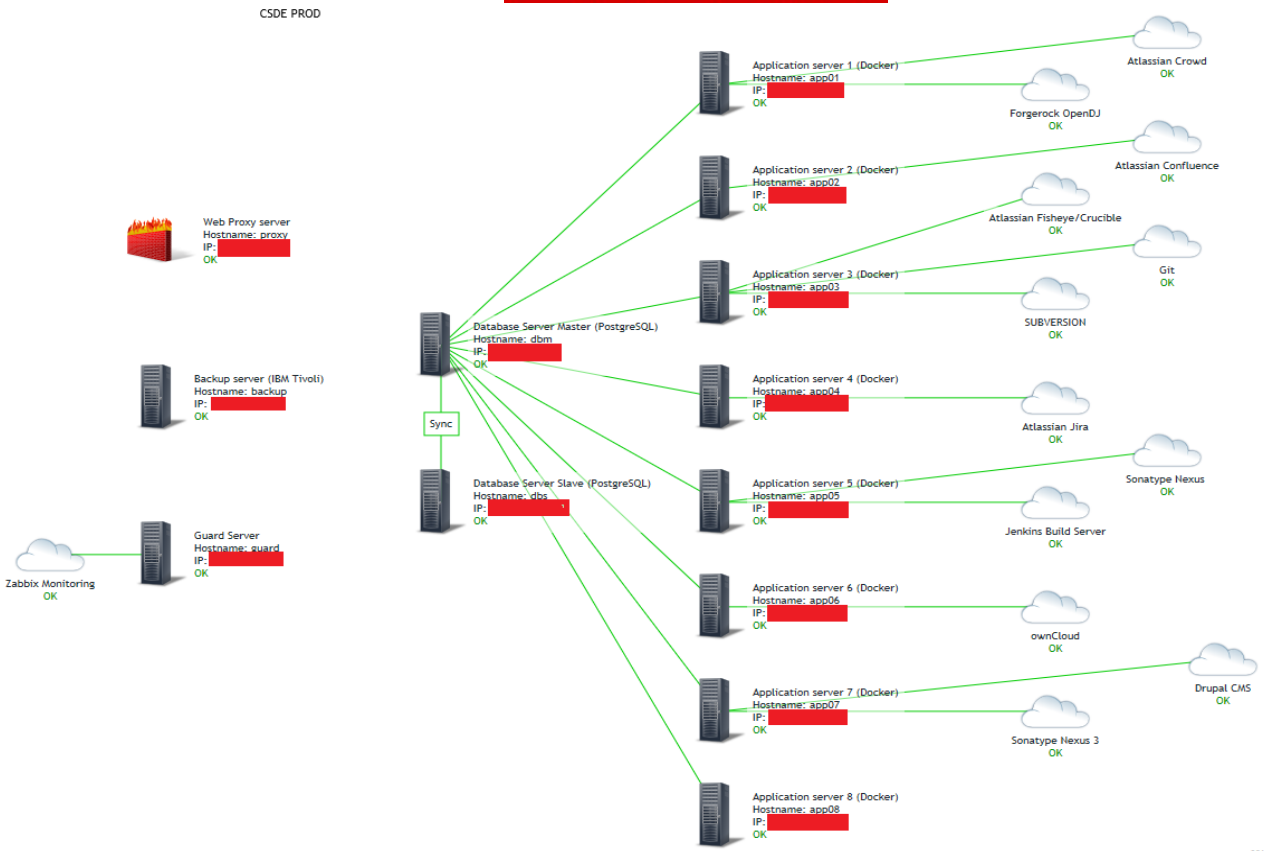
ZABBIX

- Can visualize and compare any value it monitors
- System "templates"
- Monitoring of log files and reboots *
- Local monitoring proxies **
- Customizable dashboard screens
- Real-time SLA reporting

** Albeit log and reboot monitoring means that one gets an "ERROR" and an "RECOVERY" message instead of one "CHANGED" or "REBOOTED" message. One gets used to it.*

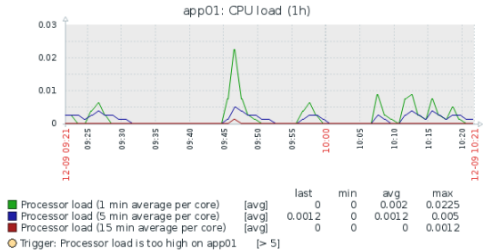
*** For example, when there are multiple sites, each site can have its own "proxy" (local Zabbix monitor), taking load off the main Zabbix server, and collecting data even if the connection to the main server is severed.*

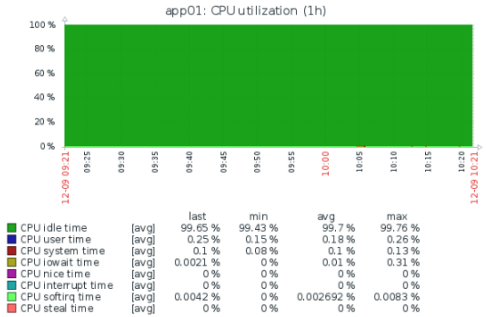
ZABBIX

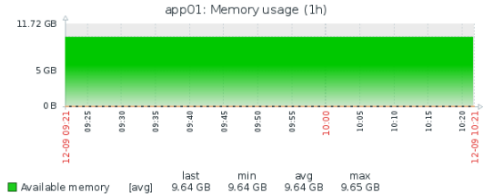

CSDE PROD
Web Proxy server
Hostname: proxy
IP:
OK
Backup server (IBM Tivoli)
Hostname: backup
IP:
OK
Guard Server
Hostname: guard
IP:
OK
Zabbix Monitoring
OK
Database Server Master (PostgreSQL)
Hostname: dbm
IP:
OK
Sync
Database Server Slave (PostgreSQL)
Hostname: dbs
IP:
OK
Application server 1 (Docker)
Hostname: app01
IP:
OK
Application server 2 (Docker)
Hostname: app02
IP:
OK
Application server 3 (Docker)
Hostname: app03
IP:
OK
Application server 4 (Docker)
Hostname: app04
IP:
OK
Application server 5 (Docker)
Hostname: app05
IP:
OK
Application server 6 (Docker)
Hostname: app06
IP:
OK
Application server 7 (Docker)
Hostname: app07
IP:
OK
Application server 8 (Docker)
Hostname: app08
IP:
OK
Atlassian Crowd
OK
Forgerock OpenDJ
OK
Atlassian Confluence
OK
Atlassian Fisheye/Crucible
OK
Git
OK
SUBVERSION
OK
Atlassian Jira
OK
Sonatype Nexus
OK
Jenkins Build Server
OK
ownCloud
OK
Drupal CMS
OK
Sonatype Nexus 3
OK

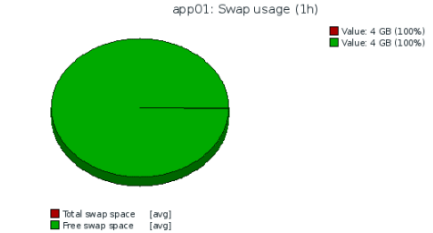
ITC GROUP

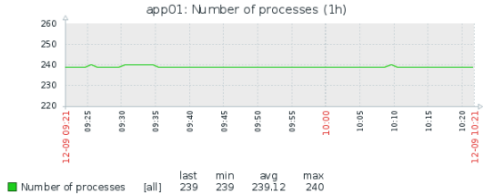
ZABBIX

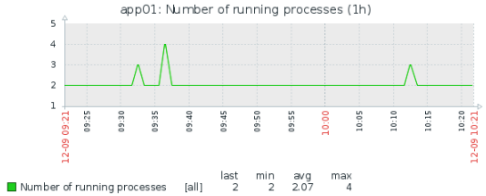












Brief info

Graylog is a free and open-source log management platform that supports in-depth log collection and analysis. Used in Network Security, IT Ops and DevOps, we can count on Graylog's ability to discern any potential risks to security, lets us follow compliance rules, and helps to understand the root cause of any particular error or problem that our apps are experiencing.

Key Features:

- Enrich and parse logs using a comprehensive processing algorithm.
- Search through unlimited amounts of data to find what you need.
- Custom dashboards for visual output of log data and queries.
- Custom alerts and triggers to monitor any data failures.
- Centralized management system for team members.
- Custom permission management for users and their roles.

Brief info

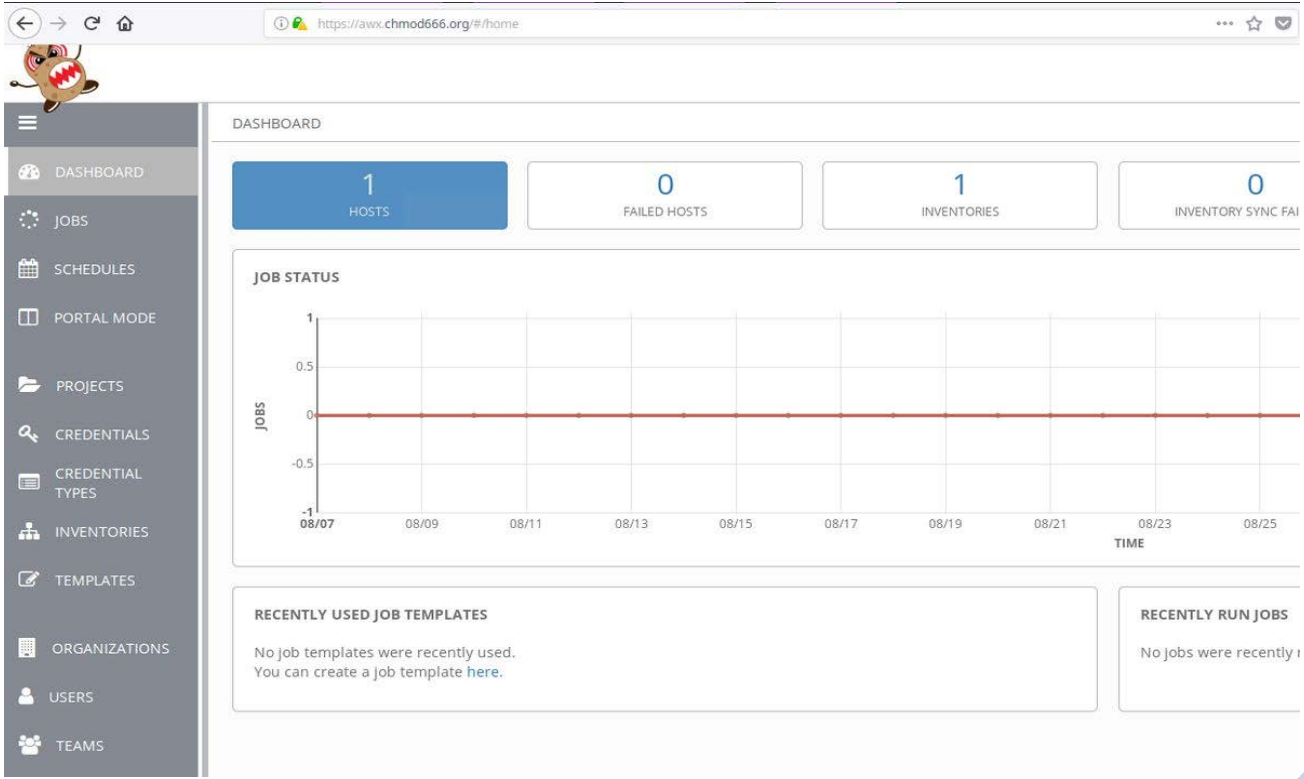
Despite the popularity of Docker, collecting and analyzing log data from Docker containers has been a challenge.

The Graylog logging driver for Docker provides automated log collection, correlation, management and analysis of Docker containers and the applications hosted on them.

Graylog's logging driver for Docker delivers the following capabilities and benefits:

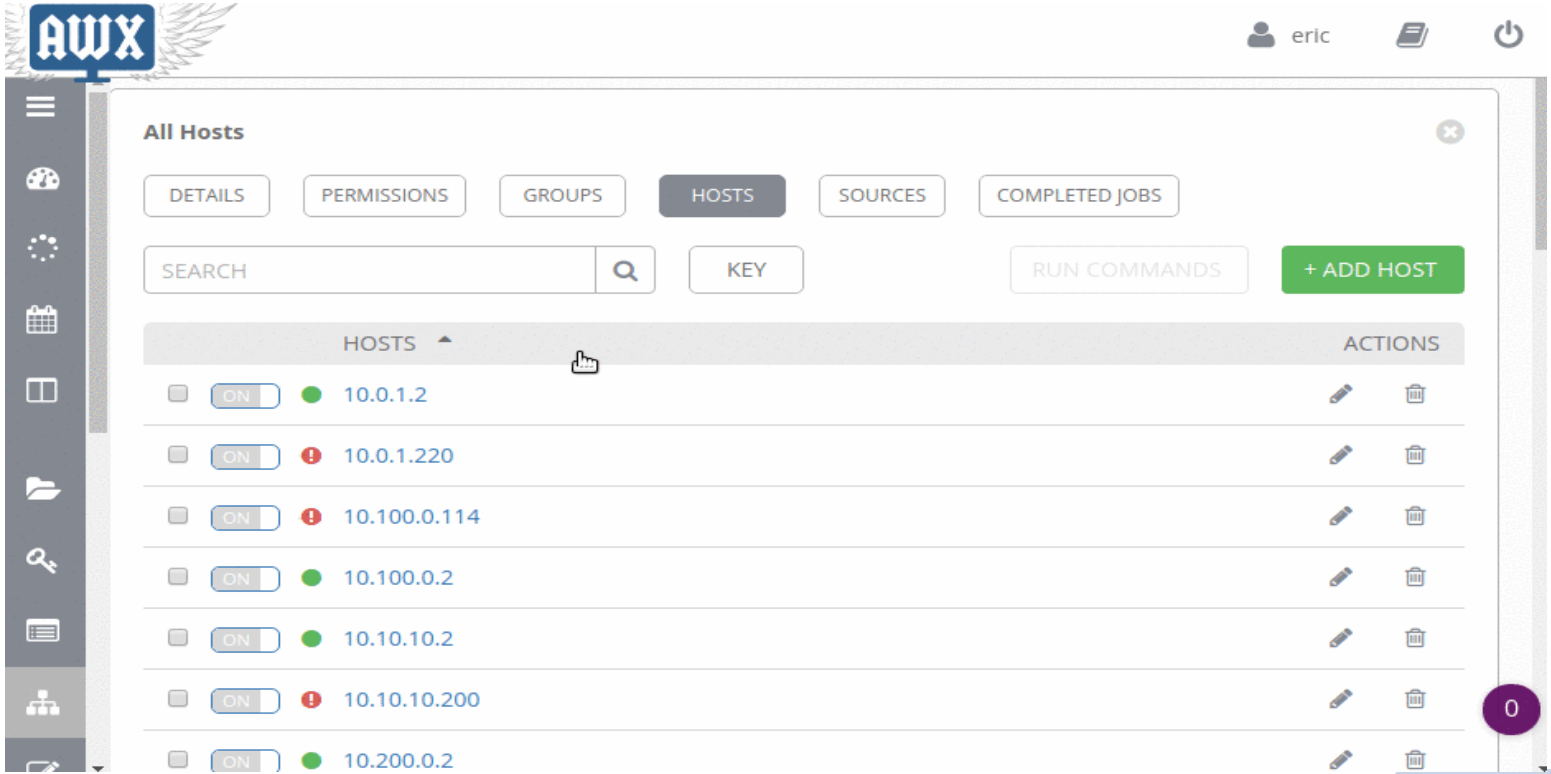
- Comprehensive application logging. Automatically collects logs from Docker containers and applications and forwards them to a central Graylog server for easy processing, storage and analysis.
- Native logging driver integrated in Docker core project. Graylog has contributed a native GELF log sender to the Docker 1.8 core project. This logging driver allows a container to send STDOUT and STDERR messages to a remote GELF endpoint like a Graylog server, making it simple to centralize Docker container logs.
- Additional application container information is also logged and sent automatically.

Ansible AWX


https://awx.chmod666.org/#/home
DASHBOARD
DASHBOARD
JOBS
SCHEDULES
PORTAL MODE
PROJECTS
CREDENTIALS
CREDENTIAL TYPES
INVENTORIES
TEMPLATES
ORGANIZATIONS
USERS
TEAMS
1
HOSTS
0
FAILED HOSTS
1
INVENTORIES
0
INVENTORY SYNC FAI
JOB STATUS
JOBS
1
0.5
0
-0.5
-1
08/07
08/09
08/11
08/13
08/15
08/17
08/19
08/21
08/23
08/25
TIME
RECENTLY USED JOB TEMPLATES
No job templates were recently used.
You can create a job template here.
RECENTLY RUN JOBS
No jobs were recently

Ansible AWX

RED HAT® ANSIBLE® Tower



Ansible AWX

RED HAT® ANSIBLE® Tower

AWX is the open source project behind Red Hat® Ansible® Tower.

AWX is built to run on top of the Ansible project, enhancing the already powerful automation engine. AWX adds a web-based user interface, job scheduling, inventory management, reporting, workflow automation, credential sharing, and tooling to enable delegation.

Here are 4 things we can do with AWX:

Delegate

Central to AWX is the ability to create users, and group them into teams. We can then assign access and rules to inventory, credentials, and playbooks at an individual level or team level. This makes it possible to setup push-button access to complex automation, and control who can use it, and where they can run it.

RED HAT® ANSIBLE® Tower

Access Reporting

Now that we have users, we'll need tools for tracking playbook runs and troubleshooting problems. AWX adds a custom callback plugin to Ansible playbook runs that captures event and output data in real time. A high-level summary of the stored output is presented on a dashboard, providing an overview of job executions, failures, and successes, as well as a breakdown of inventory successes and failures. We'll immediately know what playbooks have run and any trouble spots needing attention, and we can immediately click into the details.

At the playbook level, we'll be able to access the results and output down to an individual task on a specific host. So if that new environment the development team attempted to create didn't provision exactly as expected, we can quickly troubleshoot, and correct the problem.

RED HAT® ANSIBLE® Tower

Schedule Workflow

We'll bring together credentials, playbooks and inventory by creating a job template. The template represents the command line execution of 'ansible-playbook,' except we no longer need to touch the command line. Instead, we can run the job template on demand, and watch real-time playbook output in our web browser. Or, schedule it to run later, once or on a recurring basis, and get full access to the output whenever we need it.

In addition to running a single playbook, we can create a job template with the workflow editor that chains together multiple playbook runs. And again run it on demand, or schedule it in the future once or on a recurring basis.

Automate Through an API

At the heart of AWX is a powerful restful API. In fact, the user interface communicates with the backend entirely through the API.

The Tower CLI can also be used to make calls directly to the API from a shell script. It alleviates the need to make API calls using lower level tools like *curl* or *wget*, and provides a further source of examples, and documentation. It‘s written in Python, so the source code is a how-to guide for programming API calls.

With full access to the API, and help from the Tower CLI, it’s possible to integrate AWX with other applications like ticketing tools, source code management, continuous integration systems, and shell scripts. It gives us the ability to easily automate deployment, maintenance, and mitigation tasks that previously would have been difficult, if not impossible, with just an Ansible Playbook.

Let us consider in more detail how this could be done.

We preliminary created a template for the basic virtual machine and placed it in a VMWare datastore.

Then Ansible begins its activity.

First, the "base" role is used, which creates a virtual machine from the template and turning it on.

Next, the host name is assigned to the VM, "docker host" role implemented and the Zabbix agent is installed

About future improvements

Further, depending on which VM is created, the following roles are selected and applied:

- "postgresql master/slave" roles - on the master and slave DB servers
- "application" role - to install the container with the necessary application(Atlassian Jira, Fisheye/Crucible, Crowd etc.)
- "security" role - to implement security restrictions to container in accordance with the CIS recommendations
- "data-transfer" role - for initial and subsequent incremental data synchronization between "developer" and "production" environments

3

ABOUT US

ABOUT US

Let me introduce our company – IT Consulting Group B. V.
This is a small but very progressive firm with a wide range of services.
The registry of our services includes, but not limited to:

- IT Consulting/Audit
- DevOps
- IT Security + Data Protection
- Servers/Network Installation/Upgrade
- Storage
- Backup and Disaster Recovery
- Cloud Computing & Virtualization

ABOUT US

We are committed to deliver a faster and scalable solution for your business

For example.

What can you benefit from our virtualization & cloud computing services?

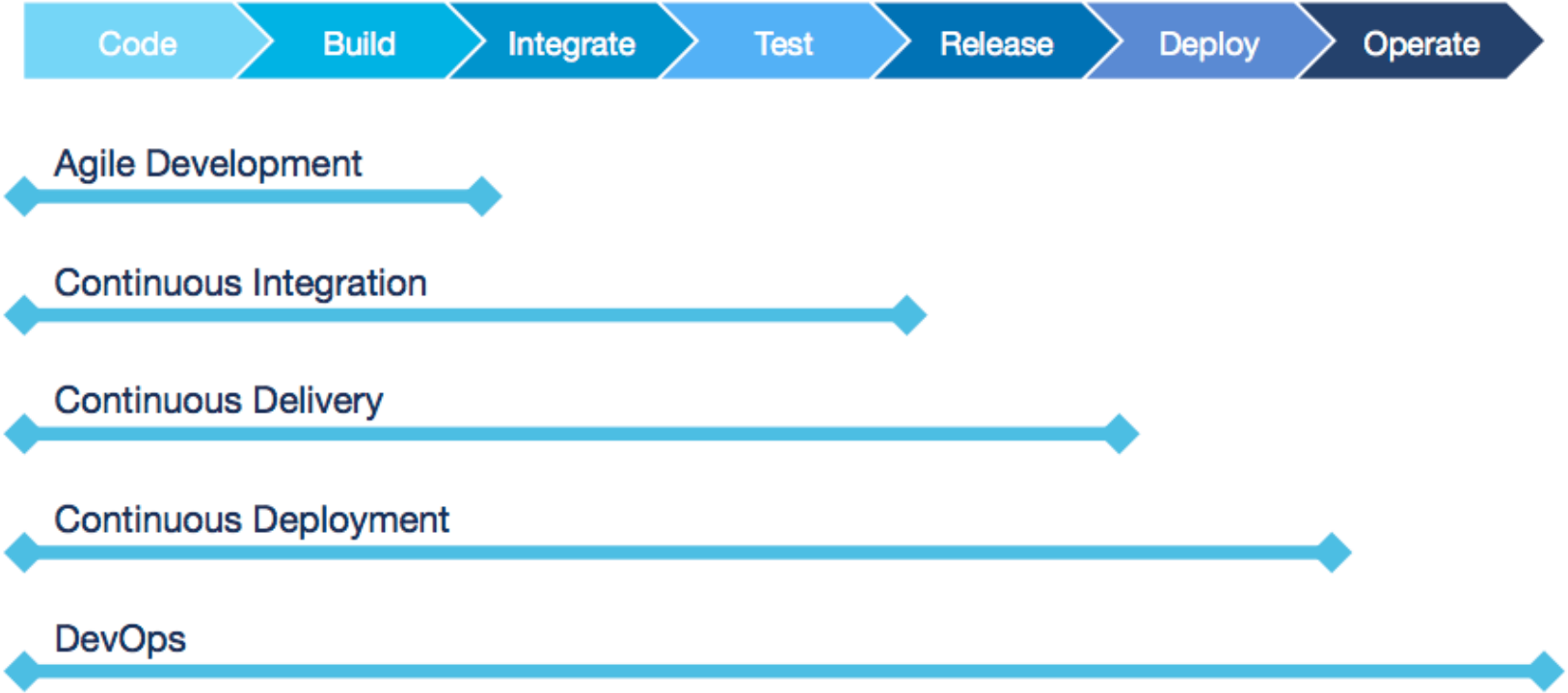
- ✓ Better flexibility
- ✓ Better resource management
- ✓ Increased security
- ✓ Improved protection against disasters
- ✓ 99.99 % uptime
- ✓ More efficiency
- ✓ Save more time and money

7Cs of DevOps

We establish the right process and then choose the right tools to deliver high quality software quickly. Below you could to find out our 7Cs approach to achieve continuous delivery:

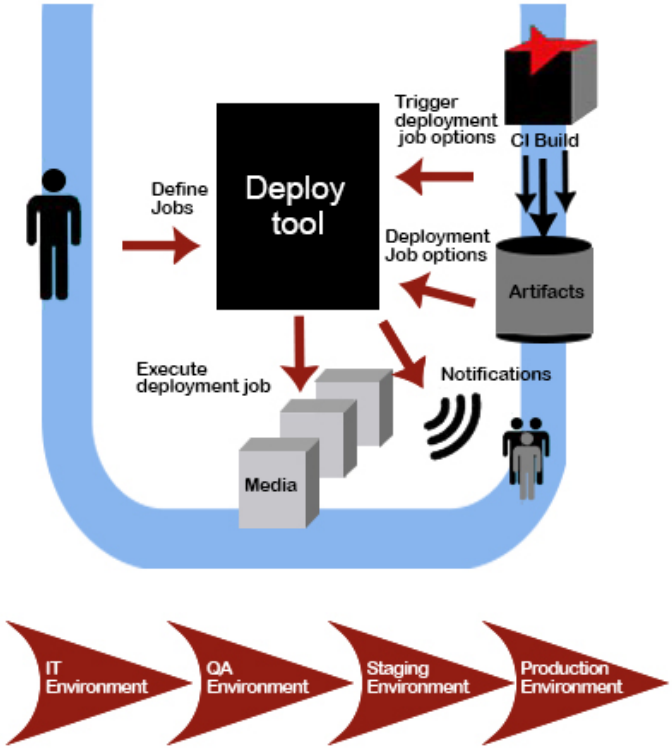
- ✓ Continuous Planning
- ✓ Continuous Development
- ✓ Continuous Integration
- ✓ Continuous Deployment
- ✓ Continuous Testing
- ✓ Continuous Feedback
- ✓ Continuous Delivery & Monitoring

ABOUT US


Code
Build
Integrate
Test
Release
Deploy
Operate
Agile Development
Continuous Integration
Continuous Delivery
Continuous Deployment
DevOps

ABOUT US

- Application Deployment
- QA Test Integration
- Deployment workflow
- Configuration
- Physical Or Virtual
- On Premises Or Cloud



- Automated Error-Free Faster Deployments
- Single Click or Continuous Deployments
- Single Tool Deploying In All Environments
- Deployment Metrics

THANKS!

Any questions?

You can find us at

https://itcgr.com & support@itcgr.com