Exploitation of DevOps concepts for the ASDEX Upgrade DCS

B. Sieglin[a], T. Zehetbauer[a], A. Lenz[a], M. Kölbl[a], A. Gräter[a], W. Treutterer[a], ASDEX Upgrade Team[a]

[a]*Max-Planck-Institute for Plasma Physics, Boltzmannstr. 2, D-85748 Garching*

Abstract

The ASDEX Upgrade Discharge Control System (DCS) is designed to be distributed and highly modular. By nature it is a complex system that requires expert knowledge to develop, deploy and operate. This paper discusses the application of DevOps concepts to DCS and its impact on the operation of ASDEX Upgrade.

The aim is to provide stable and undisturbed operation of ASDEX Upgrade, whilst having a low hurdle to include new capabilities into DCS to extend the operational scope. To ensure both goals are met DevOps concepts have been introduced to DCS. For the code development a review process was added using the code review tool Gerrit. Each change is only accepted into the code repository once the review and automated testing have been completed.

The automated testing and packaging of DCS are conducted using the automation server Jenkins. DCS is not directly installed on the host computers, but is operated using the containerization framework Docker. The containers allow easy deployment of DCS onto new systems as well as the quick change of DCS versions on the systems. A new or updated component can easily be tested by deploying the appropriate container. In case of failure the previous state is restored by redeploying the container in the previous version.

DCS components can be classified into two types, the real time and the non real time parts. Non real time parts are the services that are required to manage the configuration of the system and the setup of the real-time interaction between participants. The new deployment method with Docker is used for both non real time as well as real time services and is operational on ASDEX Upgrade since the 2019/2020 campaign. No impact of the determinism of the real time components have been observed when operated within a docker container.

1. Introduction

Operation of fusion devices is a complex and challenging task. In case of a tokamak, such as ASDEX Upgrade, plasma operation requires sophisticated, reliable and fast control. Considering that the aim of current fusion devices is to conduct experiments to enable future fusion power plants, the control system has to be flexible and extensible in order to support the increasing demand for advanced control. This flexibility and extensibility is increasing the difficulty to achieve reliability. In order to ensure both for ASDEX Upgrade, DevOps principles stemming from software development have been applied to the development and deployment of the discharge control system (DCS). One requirement of the implementation is the use of *standard* solutions which have a wide user base. This is done to reduce the work required for the setup and maintenance of the system. In section 2 of this paper the DevOps principles and their implementation are discussed. The development and deployment strategy is discussed in section 3.

2. DevOps

The term DevOps is a combination of *software development* Dev and *IT operations* Ops. It denotes a set of practices that aim to reduce the system development time and provide continuous delivery with a high level of software quality.

A central instrument to implement the DevOps practices is the so called *toolchain*. The function of the toolchain is to aid the development, delivery and management of software applications throughout the system development life cycle. The toolchain itself consists of multiple tools and services that interact in order to fulfill the desired tasks.

1. Code development and review, source code management and coordination of collaborative working.
2. Building the software and monitoring the current build status.
3. Testing the software and identifying issues as soon as possible.
4. Packaging the software and pre-deployment staging.
5. Releasing the software for deployment.
6. Configuration and management of infrastructure.
7. Monitoring of the production systems.

It has to be noted that the toolchain described here contains only the functional parts from coding to deployment on the production systems. Tools for change management and extensive monitoring are not discussed in this paper but are currently in different states of implementation on ASDEX Upgrade. On ASDEX Upgrade the

Email address: bernhard.sieglin@ipp.mpg.de (B. Sieglin)

toolchain is using the collaboration tool Gerrit [1] for hosting of the source code and configuration as well as for code review. Each code change made in DCS is submitted to Gerrit which starts a code review before the change is accepted into the repository. Only after the approval of the change by at least one responsible officer can the change be submitted to the code repository.

For workflow based task execution the automation server Jenkins [2] is used. This service is the backbone of the toolchain performing all tasks that are required during the whole development and deployment process. Jenkins builds and tests the changes submitted to Gerrit, packages the software and deploys it onto the different systems. For this so called *pipelines* are defined which perform the different tasks and report the results. This is used e.g. by Gerrit to judge if a code change passed the defined tests. Once a change is pushed to Gerrit different pipelines start their tasks on this change, e.g. building packages, performing unit tests or linting of the code and the results are collected. Together with an approved review the successful pass of all tasks is a prerequisite for change submission into the code repository. In case of failure the submission is blocked. Note here, that a responsible officer can, if required, overwrite the blockage, but has to do so explicitly. This can be necessary if for example a change can only work if changes of multiple projects are submitted together.

For configuration management and system orchestration Ansible [3] is used. Using Ansible the deployment and orchestration process becomes a configuration task. It allows the definition of logical roles of components within the system, including dependencies and inclusion of other roles. For the deployment process on each platform so called *playbooks* are defined which combine multiple roles to define the capabilities of the system.

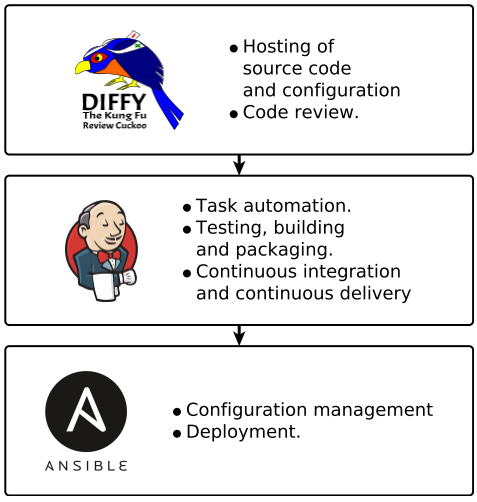

Figure 1: Illustration of the components of the toolchain employed on ASDEX Upgrade.

This toolchain setup ensures an automated and reproducible development and deployment process in which the usage is unified and common mistakes, like e.g. forgetting a step during deployment, are eliminated. In addition to the tasks that are performed for each change there are tasks that are performed on a regular basis. For example the latest state of DCS is build completely every night. This is done to see if any change has a breaking impact on other parts of DCS. Although the toolchain introduces some overhead during setup and usage its benefits have outweighed this additional effort during the period of its application to the last to ASDEX Upgrade experimental campaigns from 2018-2020.

3. Development and Deployment

The aim of DCS is to provide a stable environment for the operation of ASDEX Upgrade, whilst enabling enough flexibility to implement and test new control schemes / scenarios / experiments. In order to ensure stable operation of ASDEX Upgrade the development and deployment process has been formalized.

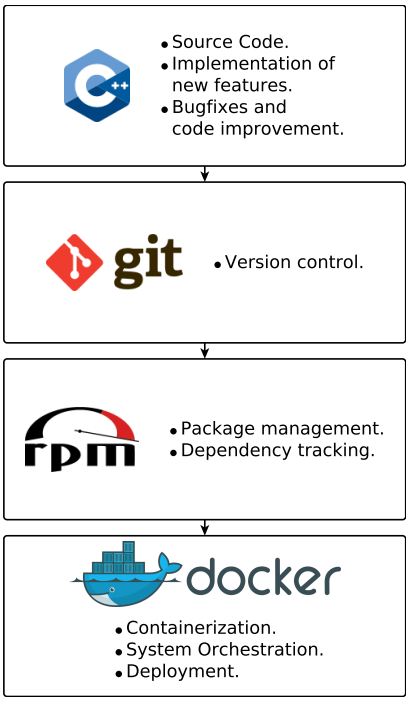

Figure 2: Illustration of the technologies employed for the development and deployment process in ASDEX Upgrade.

For version control of the source code the distributed version control system git [4] is used. One reason for this is that Gerrit is based on git, another is that git is commonly used in software development and therefore has a wide support. The packaging is done using RPM [5] since CentOS [6] and RedHawk [7] is used as the operating system. RedHawk is used for systems where deterministic timing is required, CentOS is used for systems where this is not required. This provides the dependency tracking to ensure all components required for a certain service are

installed properly. Docker [8] is used to containerize the different services required to operate DCS. Docker is an OS-level virtualization with which software is delivered in packages called containers. Each container bundles its own software, libraries and configuration and is isolated from other containers. The containerization enables modular orchestration of the running systems with versioning of the source code, the packages and the running system. The different containers communicate to each other using well-defined channels. The containers run on the host operating system kernel. This uses less resources compared to virtual machines, which will become important in section 4 where the application to both non real-time as well as real-time processes is discussed.

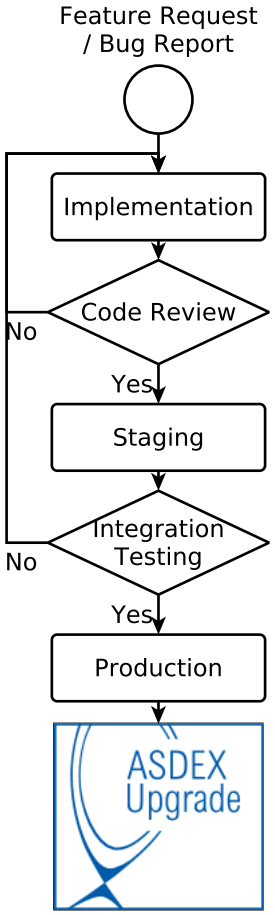

Figure 3: Illustration of the development and deployment process employed for the ASDEX Upgrade DCS from implementation to production.

In addition to the technical development and deployment process, an additional separation between staging and production has been introduced. Once a new feature is implemented and has successfully passed the code review it does not immediately go into production but it goes into staging. For staging, as well as for production, the whole deployment process is undertaken up to the point that the final docker images are built. This is done since a change, even if it is syntactically and logically correct, might break other parts of DCS. Staging enables testing of new features without already releasing the changes for production use. This allows to test the interaction of the change with the other components of DCS. Once the changes have been tested they are released into production and new production docker images are built to be used in operation.

4. DCS Services

The DCS services consist of two main types, real-time and non real-time services. For each service a separate docker image is created. The docker image for the real time services contains all application processes (AP) available in DCS. During the deployment of the docker container the configuration of the container is set. This determines which DCS configuration is loaded during operation, defining which APs are instantiated for the specific monitoring and control tasks. The orchestration of the running system is done using docker compose. This allows the specification of the different services and dependencies via a YAML configuration file. In addition the communication channels, e.g. ethernet ports, are specified and resources, like CPU affinity, available memory, etc can be assigned. For the real time services the corresponding capabilities have to be enabled in both the kernel and the configuration of the docker daemon. In the docker compose file of the real time services the permissions have to be granted to allow operation with real time policies. During the commissioning of the docker based deployment, as well as during operation, the performance of the real time system has been monitored. All real time systems have an in house developed Time-to-Digital-Converter (TDC) [9] installed which provides time synchronisation between the systems. These provide a time reference with a resolution of 20 ns and are used to monitor the timing of the DCS real time processes. The jitter in the timing of the DCS cycles has been compared for the same software, directly installed on the system and run using the docker image. No difference in the timing has been observed and the jitter was in the order of around 5 μs for both variants. This is similar to the jitter determined from cyclic tests of the underlying system.

In order to have an easy to use user interface for the orchestration of the different systems and services Portainer [10] is used. This provides a web interface which allows configuration, deployment and monitoring of multiple systems. Using this starting, stopping or deploying DCS is literally done by the push of button. During operation this proved to be comfortable, reducing the workload and effort of the DCS operator.

The encapsulation of the different services into separate docker containers enables the heterogeneous deployment of services onto staging or production machines. This means that, as long as the communication protocol between the different services is not changed, images from staging and production can be operated together. Note, however, that for a control system, the compatibility of communication

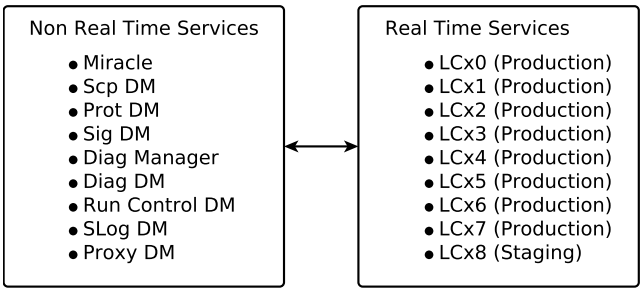

Figure 4: Illustration of the services forming the core DCS on ASDEX Upgrade. Note: LCx8 is dedicated for integration tests of staging on the production system.

protocols is only one prerequisite for operating production and staging images together. They must also be compatible semantic-wise. I.e. the staging image under test must have no or only a well-confined influence on the control system actions.

The possibility to operate a heterogeneous system, containing of staging and production images, increases testability of new features whilst decreasing the impact of errors. In case an error is detected a rollback to the previous operational version is done by redeploying the previous configuration. This capability has proofed to be very valuable. Considering the number of people required to operate a fusion device and the costs associated with this it is, to say the least, reassuring to be able to introduce a change into the control system without the danger of preventing operation. The common practice employed at ASDEX Upgrade is not to change the deployed control system on operation days. During an ASDEX Upgrade campaign there are typically five operation days within two weeks. This leaves time in between two operation days to deploy an updated control system version and conduct tests on the production environment. Should however, despite all the testing done before, an error occur during operation due to the newly deployed control system version, it is now always possible to rollback to the previous working version within a couple of minutes. The docker based deployment has been employed since the 2019-2020 campaign, before DCS was deployed by installing the RPM packages directly onto the target systems. The new deployment strategy has been in place since 2018 and since then no experimental time has been lost related to the development and deployment process. However, it has to be mentioned that the gained flexibility from the docker base deployment significantly decreased the hurdle to introduce new features into DCS. Examples for this are the active control of the X-point radiator and the improved actuator management.

5. Unified system setup

So far the application of DevOps principles to the DCS of ASDEX Upgrade has been discussed. This, however, is only one component which is required for the operation of ASDEX Upgrade during an experimental campaign. In addition to this there are a variety of other systems and services which have to operate. An example for this are real time diagnostics which provide data for control during plasma discharges. The development and deployment strategies discussed for the DCS have also been applied to the so called DCS satellites [11]. Operation of DCS satellites with and without data acquisition hardware has been successfully demonstrated. The real time part of the in house build SIO2 data acquisition system [12] has been successfully operated using the docker based deployment. Utilizing this deployment method helps to reduce the effort required for the initial setup as well as for maintenance. Starting from the 2020-2021 campaign the base system for the real time DCS systems will be unified. It will consists of a bare real time Linux (CentOS or RedHawk) installation with all hardware drivers and a running docker daemon. The purpose of each system will only be determined by the hardware that is installed in the system, e.g. IO hardware or data acquisition and the software that is deployed on the system using docker.

6. Conclusions

Employing DevOps concepts on ASDEX Upgrade has proven to be successful. It has been observed that the overall reliability of DCS operation has improved. The development speed and quality has increased whilst achieving a reduction in the risk of introducing errors into the production environment. All together this ensured a stable environment for the ASDEX Upgrade experimental campaigns. During the experimental campaigns of ASDEX Upgrade from 2018-2020 no experimental time was lost due to DCS errors related to the development and deployment process.

It is foreseen to further extend the DevOps concepts to e.g. change management and monitoring. The current state achieved on ASDEX Upgrade proved the viability of DevOps. However, one has to consider the limited available resources for operation and development. Taking this into account the efficiency has to be improved continuously. This issue is not limited to fusion research but is a common problem encountered in most industries. Therefore solutions for these problems are under constant development by a large community, which can be readily adapted and utilized.

Acknowledgement

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 and 2019-2020 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

References

[1] Gerrit code review (2020).
URL https://www.gerritcodereview.com/

[2] Jenkins (2020).
URL https://www.jenkins.io/

[3] Ansible (2020).
URL https://www.ansible.com/

[4] Git (2020).
URL https://git-scm.com/

[5] Rpm package manager (2020).
URL http://rpm.org/

[6] Centos (2020).
URL https://www.centos.org/

[7] Redhawk linux (2020).
URL https://www.concurrent-rt.com/products/redhawk-linux/

[8] Docker (2020).
URL https://www.docker.com/

[9] G. Raupp, R. Cole, K. Behler, M. Fitzek, P. Heimann, A. Lohs, K. Lüddecke, G. Neu, J. Schacht, W. Treutterer, D. Zasche, T. Zehetbauer, M. Zilker, A "universal time" system for asdex upgrade, Fusion Engineering and Design 66-68 (2003) 947 – 951, 22nd Symposium on Fusion Technology.

[10] Portainer (2020).
URL https://www.portainer.io/

[11] B. Sieglin, W. Treutterer, L. Giannone, Dcs satellite: Enhanced plant system integration on asdex upgrade, Fusion Engineering and Design 146 (2019) 1737 – 1740, sI:SOFT-30. doi:https://doi.org/10.1016/j.fusengdes.2019.03.028.

[12] K. Behler, H. Eixenberger, B. Kurzan, A. Lohs, K. Lüddecke, M. Maraschek, R. Merkel, G. Raupp, G. Sellmair, B. Sieglin, W. Treutterer, Recent diagnostic developments at asdex upgrade with the fpga implemented serial i/o system "sio2" and "pipe2" daq periphery, Fusion Engineering and Design 159 (2020) 111873.