7

Deployment Tools

The deployment specification should define execution architecture of systems that represent the assignment (deployment) of software artifacts (i3-MARKET building blocks) to deployment targets (usually nodes).

Nodes represent either hardware devices or software execution environments. They could be connected through communication paths to create network systems of arbitrary complexity. Artifacts represent concrete elements in the physical architecture.

Once the deployment has been provided, a complementary specification would be necessary to define how to deploy software within the i3-MARKET ecosystem. In the context of i3-MARKET, we will be referring to this specification as management operative specification.

This chapter gives guidance on how the solutions for deploying i3-MARKET software are defined within the i3-MARKET instances as part of the deployment operative. The i3-MARKET operative considers four possible deployment scenarios categorized as manual or automated deployments and oriented towards i3-MARKET developers and/or data spaces and/or data marketplaces infrastructure administrators.

For the deployment and management operative, Ansible and Zabbix have been proposed as configuration, management, and monitoring tools, respectively, for the central environment. It is left to the stakeholders to decide which tools will be used and deployed for managing and monitoring the marketplace instances.

7.1 Solution Design

A four-layer stack has been defined for i3-MARKET: at the lowest layer, there is the Cloud provisioning and management layer (Figure 82). On top of that, a DevOps software layer is placed for assembling all the software used for the CI/CD process. Then, a third-party software layer is in charge of giving

support to the i3M-core artifacts, which can be found at the top level of the stack.

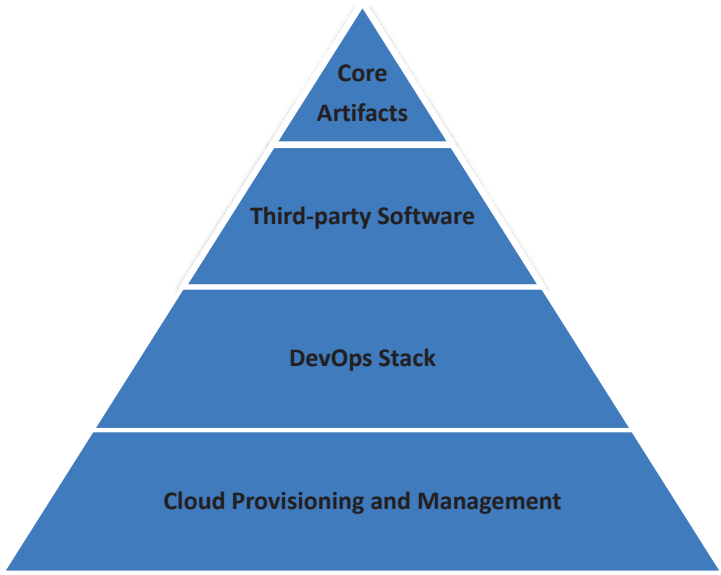

Figure 7.1 Four-layer i3M SW stack.

Depending on the environment to be deployed, it might be deployed on one layer or another. More details on the specific software deployed on each environment are given in the following sub-sections.

The target audience are the i3-MARKET project developers who are participating in the development and deployment of the i3-MARKET Backplane.

The i3-MARKET operative considers four possible deployment scenarios, categorized into manual and automatized deployments. These scenarios are the following:

- Manual deployment scenario one (MDS1)
- Automatized deployment scenario with Ansible (ADS1)
- Automatized deployment scenario with Ansible and GitHub CI/CD (ADS2)
- Automatized deployment scenario with Docker Compose (ADS3)

Considering an i3-MARKET user role perspective, the main roles involved in the different deployment scenarios are:

- i3M root instance admin

- i3M SW developer
- i3M third-party SW admin
- i3M pilot instance admin

Table 7.1 provides the mapping between the i3-MARKET user roles and the previously listed deployment scenarios:

Deployment scenario/user role	I3m root instance admin	i3M SW developer	i3M third-party SW admin	i3M pilot instance admin
MDS1	✗	✓	✓	✓
ADS1	✓	✗	✓	✓
ADS2	✗	✓	✗	✓
ADS3	✓	✓	✗	✓

Table 7.1 Deployment scenarios and i3M user roles mapping.

The following subsections describe in detail each identified deployment scenario.

7.1.1 MDS1: manual deployment

The manual deployment scenario one (MDS1) is based on accessing the physical resources by establishing an SSH connection. Once the physical resource is accessed, the user proceeds with the SW deployment manually. An overview of MDS1 is provided in the following picture. The actors involved in these scenarios are i3M SW developer and i3M third-party SW admin (Figure 7.2).

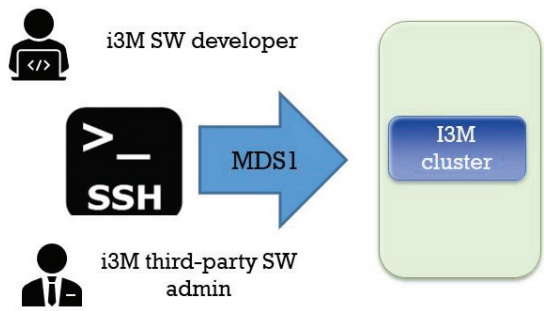

Figure 7.2 MDS1.

7.1.2 ADS1: automated deployment with Ansible

Automated deployment scenario one (ADS1) is based on the provision of a set of Ansible playbooks containing deployment recipes. Playbooks are one of the core features of Ansible and tell Ansible what to execute. They are like a to-do list for Ansible that contains a list of tasks. Playbooks contain the steps which the user wants to execute on a concrete physical resource, and they are run sequentially.

From an operative point of view, actors involved in this scenario must cover the following deployment workflow:

1) Create an Ansible template (playbook) with concrete deployment instructions using the physical resources specified.
2) Start an Ansible job by instantiating the playbook template provided in step 1.

An overview of ADS1 is provided in the following picture. The actors involved in this scenario are i3M IT admin and i3M third-party SW admin (Figure 7.3).

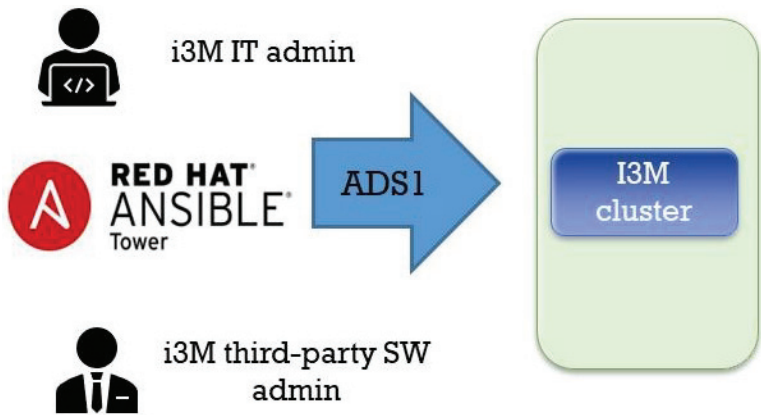

Figure 7.3 ADS1.

Finally, Figure 7.4 contains a playbook example showing the main structure in terms of tags to be included i3-MARKET playbooks, which are: name, hosts, vars, and tasks.

```
---
   name: install and configure DB
   hosts: testServer
   become: yes

   vars:
      oracle_db_port_value : 1521

   tasks:
   -name: Install the Oracle DB
      yum: <code to install the DB>

   -name: Ensure the installed service is enabled and running
   service:
      name: <your service name>
```

Figure 7.4 Ansible playbook example.

7.1.3 ADS2: automated deployment with Ansible and CI/CD GitHub pipelines

Automated deployment scenario two (ADS2) is based on the provision of CI/CD pipelines with Ansible and GitHub.

An overview of ADS2 is provided in Figure 7.5. The only actor involved in this scenario is i3M SW developer (Figure 7.5).

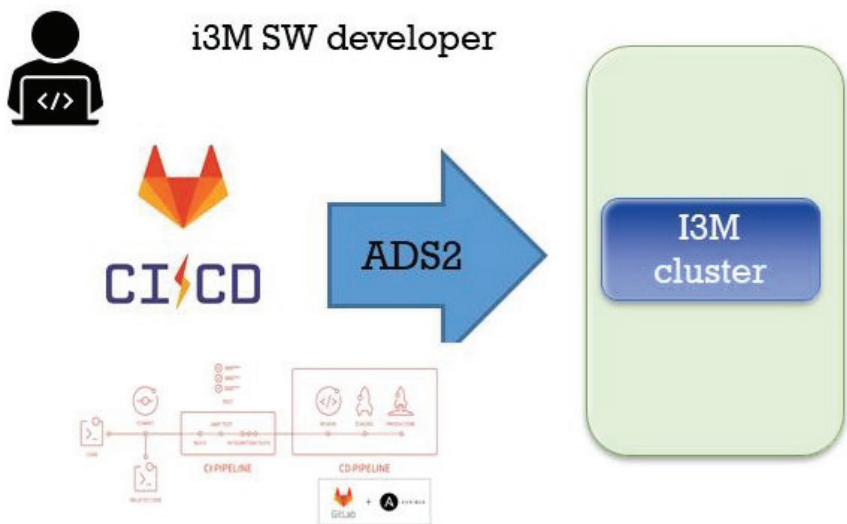

Figure 7.5 ADS2.

The goal to reach in the current deployment scenario should be aligned with i3-MARKET DevOps strategy and based on the provision of an Ansible Tower CI/CD architecture.

Considering the approach presented at the CI/CD Ansible Tower and GitHub sites [78], Figure 7.6 illustrates what we should build to support CI/CD in i3-MARKET using Ansible and GitHub.

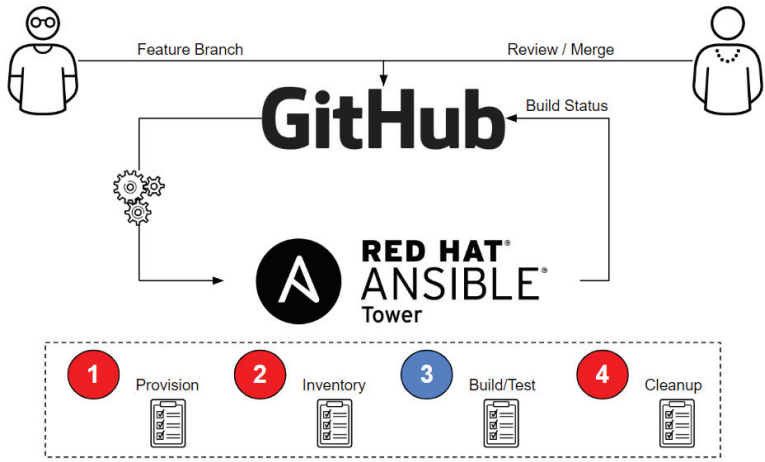

Figure 7.6 CI/CD with Ansible and GitHub.

As is well known, the main purpose of CI is of course to protect the master branch so that it always compiles. The only way to do this is to check the code in another branch (like a function branch), test that code, review the code, and only merge it with the master once all tests pass. The architecture above achieves exactly that and does so with a very simplified approach that leverages Ansible Tower as our CI engine. For the CD part, only a few additional workflows would be needed to implement artifacts generated by the CI process in dev -> test -> production. Using this architecture, one could use the GitHub versions to store artifacts. GitHub has the ability to trigger a webhook when the latest version is updated, which in turn could trigger an Ansible Tower CD workflow.

7.1.4 ADS3: automated deployment with Docker Compose

The last way of automatizing the deployments on i3-MARKET is by means of Docker Compose[1]. After the last release of the deployment strategy adopted

[1] https://docs.docker.com/compose/

by i3-MARKET of having *N* decentralized i3-MARKET instances + 1 master i3-MARKET instance for centralizing some services, a deployment for supporting the installation of an i3-MARKET instance (a decentralized node) has been created based on Docker Compose. This Docker Compose is used for deploying and managing multiple Docker containers, each of them containing different core and decentralized services developed by i3-MARKET.

This mechanism allows any marketplace to deploy an i3-MARKET "pilot environment" in order to be part and interact with the i3-MARKET ecosystem. Therefore, ADS3 becomes the most useful deployment strategy for supporting i3-MARKET pilots in the deployment of those i3-MARKET services, which need to be decentralized and installed in the pilot premises. These services are: "Backplane" (Backplane API component), "tokenizer" + "pricing-manager" (Monetization component), "sdk-ref-impl" (SDK-RI component), "web-ri" + "mongo_web-ri" (Web-RI), "oidc-provider-app" + "oidc-provider-db" (Service-centric authentication component), "vc-service" (User-centric authentication component), semantic-engine + semantic-engine-db (Semantic engine component), data_access (Data access component), auditable-accounting (Auditable accounting component), besu (Blockchain network pilot node), cockroachdb-node (Distributed storage component), conflict-resolver-service (Conflict resolution component), rating (Rating Component), and "keycloak" (Security server component).

In terms of the Docker Compose file definition, a set of "env.*component*" files has been created for storing config information relative to the deployment of each of the services contained in the Docker Compose file.

Besides installing the decentralized services by means of the Docker Compose file, the administrator of the pilot infrastructure must install a wallet.

Interaction with i3-MARKET can be done in several ways:

- By using the API of the Backplane, the SDK-RI or using the SDK-core libraries to integrate our application.
- By using the Web-RI.
- By managing an instance (pilot-side or central) of i3-MARKET. More details on this usage can be seen in the marketplace instance administration.

Marketplaces must be accepted to join the federation. Currently, the rules of the federation have been decided and are defined as part of the following section for the summary onboarding process. Once a marketplace is part of

i3-MARKET, it can issue credentials to its consumers, providers, and data owners.

7.2 i3-MARKET: Onboarding Process

This process describes the onboarding steps for installing an operative node (pilot environment) that allows a pilot being able to interact with other marketplaces inside the i3-MARKET ecosystem. It is a practical guide that makes use of the automated deployment based on Docker Compose (ADS3) commented in the previous section.

The required steps are:

1) Clone i3-MARKET deployment repository
2) Login into i3-MARKET Nexus and Git repos
3) Execute Docker Compose
4) Install i3M Wallet
   Go to Wallet[2] and download the version suitable for your operating system and do the following actions for:
   - Windows operating system:
     - Download and execute wallet desktop.
     - The application is a standalone RAR file. Extract it and execute the i3M Wallet.exe file.
   - MacOS operating system:
     - Open the dmg file and install the wallet desktop application.
   - Linux operating system:
     - For Debian-based systems, you can use the deb package:
       - # change x.x.x for the version.
       - sudo dpkg-i wallet-desktop-x.x.x-amd64.deb.
5) Create a wallet and a consumer and/or provider identity in the wallet.
   The first time a user initiates the application, a dialog asking for a password appears. The user will have to introduce this password each time the application starts.
   Create a wallet named i3-MARKET, type HD SW Wallet, and i3-MARKET network.

[2] https://github.com/i3-MARKET-V3-Public-Repository/SP3-SCGBSSW-I3mWalletMonorepo/releases

Create a Consumer and/or Provider identity (right-click over the i3-MARKET wallet).

6) Register a new OIDC client.
Access your local instance of WEB-RI (i3-MARKET GUI) available in http://localhost:5300/.
Note: The OIDC client configuration is automatically done from the WEB-RI. Just those who are interacting directly through the SDK-RI or SDK-core must do it by following the next steps:
No OIDC client registered? Please follow the below steps:

   i. Ask your i3-MARKET admin for your corresponding "i3-MARKET OpenID Connect Provider API"[3] (by default, each instance of i3-MARKET has its own provider) endpoint to get an initial token for registering a new client (authorize green button)
      - Try logging in and get *initialAccessToken*.
      - Use *initialAccessToken* as *bearerAuth*.

   ii. Then here, using the access token as bearerToken (press the lock symbol to open the form to paste the token), you can register a new client. Please note that you must add the following information:
      - http://localhost:5300/api/credential in redirect_uris field
      - http://localhost:5300/auth in post_logout_redirect_uris field

   After successful client registration, you can paste the returned information in the text area.

7) Generate credentials for the consumer/provider identity.
Start the authentication workflow from local WEB-RI instance by following next steps:

   a. Provide a username for consumer role
   b. Wallet pairing
   c. Select wallet identity
   d. Add Verifiable Credentials to the wallet
   e. Login using credentials generated previously
   f. Selective disclosure
   g. Sign
   h. Access finally to GUI of Web-RI.

[3] And endpoint similar to: https://XXXX.i3-MARKET.eu/release2/api-spec/ui/#/Developers/get_release2_developers_login