Oracle ACE Pro

Gitlab – How we built handsom pipelines

Tomasz Ziss

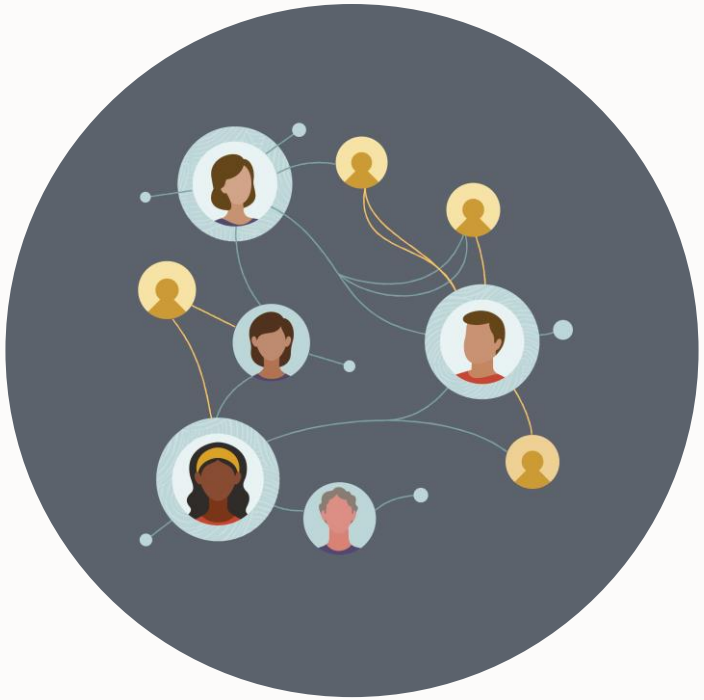
Cloud Advisory Innovation Principal Manager at
Accenture Enkitec Group

Tomasz Ziss

About Me

- ~15 years in IT industry
- From Poland, Warsaw
- Cloud Advisory Innovation Principal - Accenture Enkitec Group
- Oracle Ace Pro
- Oracle Certified Master
- Multi-cloud certifications – OCI, AWS, Azure, GCP
- Postgraduated in Data Engineering and Machine Learning
- Blogger -> https://tziss.wordpress.com

Learn more & connect with the Oracle ACE Program



Oracle ACE

Program Details	ace.oracle.com
Nomination	ace.oracle.com/nominate
ACEs in Action Blog	blogs.oracle.com/ace
X	@oracleace
Facebook	The Oracle ACE Program
Linkedin	bit.ly/OracleACEs
Email	ACEprogram_ww@oracle.com

Oracle ACE Pro

Agenda

- ❑ Gitlab Ci/CD overview
- ❑ Our automation environment
- ❑ Pipeline guided tour
- ❑ Summary – Lessons learned

Gitlab Ci/CD

General

- Complete DevOps platform
- Basic structure is already knitted differently than classic DevOps Tools
- DevOps principle of visibility of work and progress

Architecture & Maintenance

- Gitlab server + Gitlab runners
- Self-Managed Gitlab vs Gitlab as a Service
- Only ready-made integrations
- Security Vulnerabilities via patching without downtime on pipelines

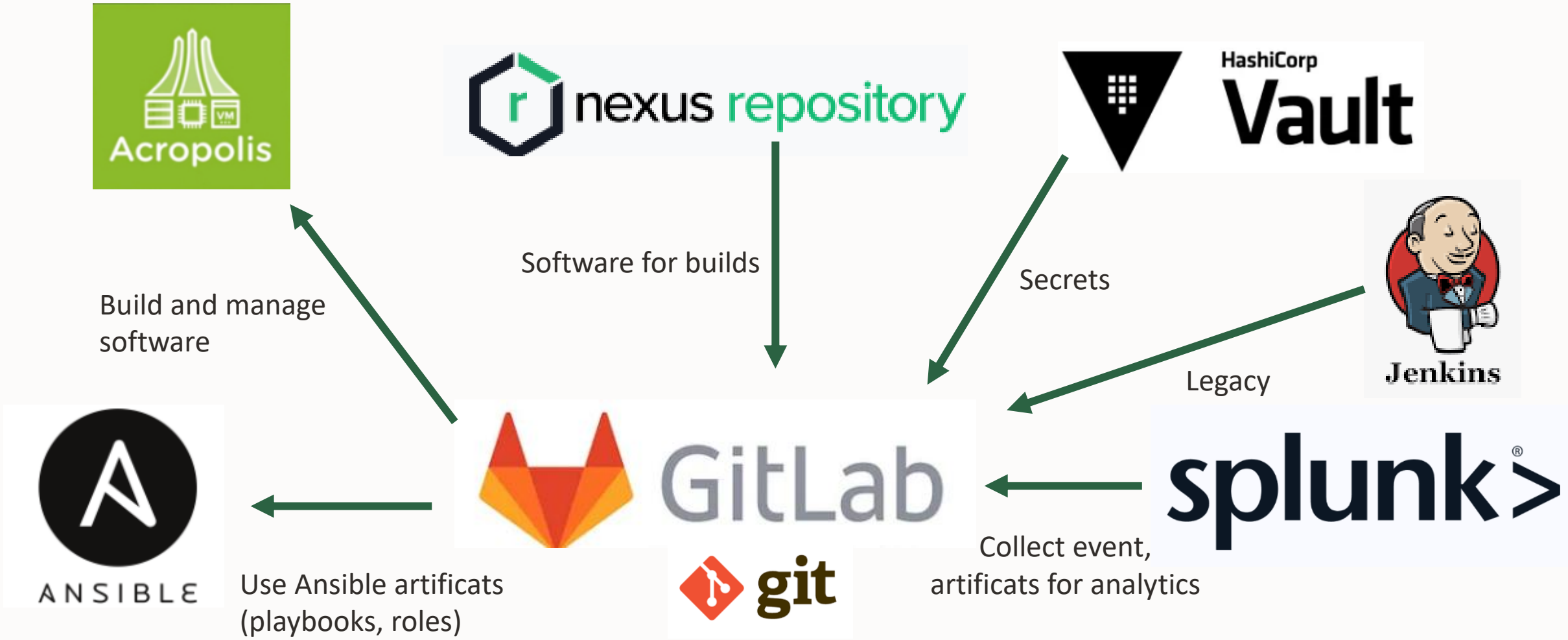
Expandability

- JIRA integration
- Seucrity scanner
- Not everything is configurable and extensible like in Jenkins. Good portability as it is similar everywhere

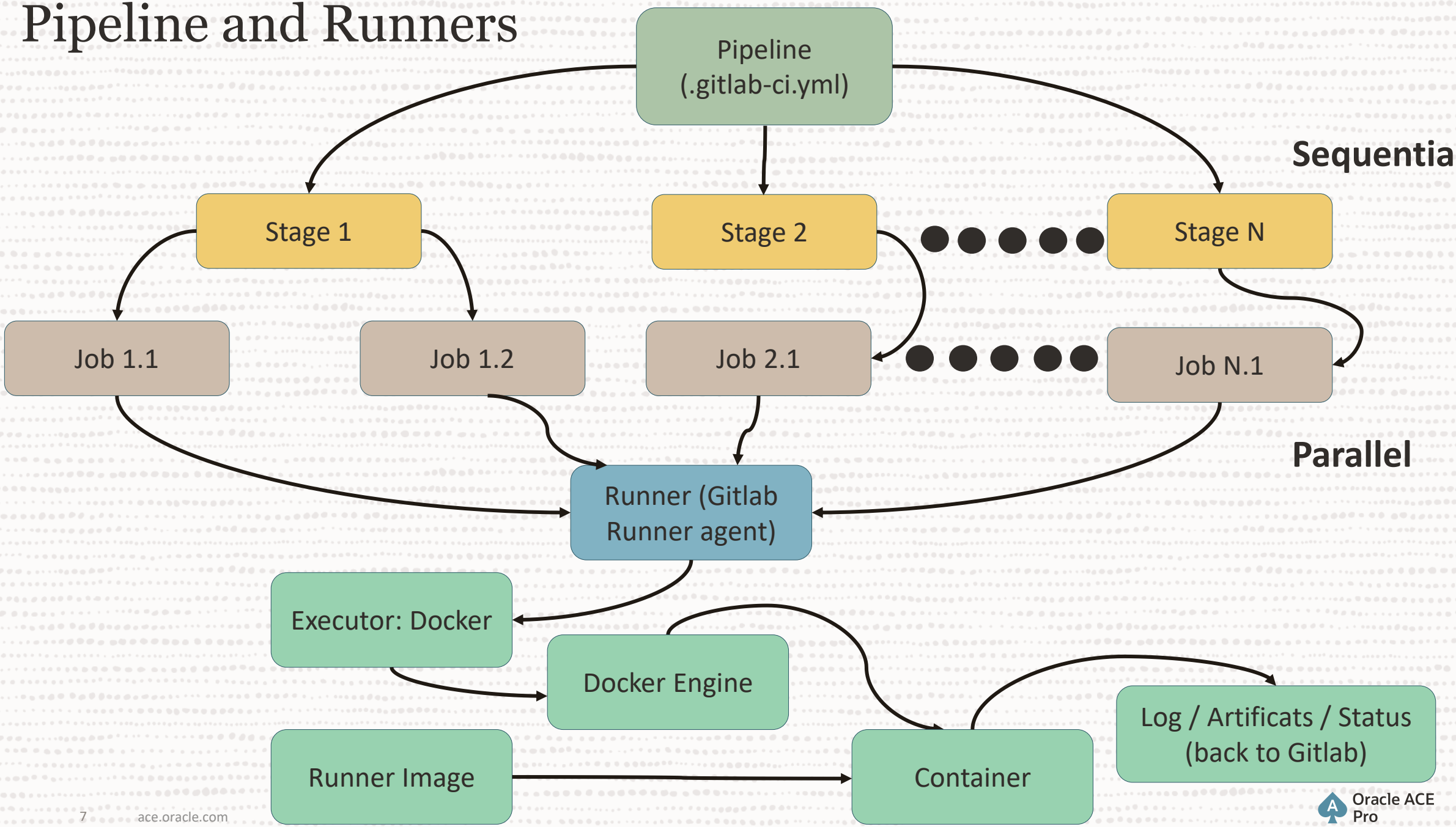
Pipelines

- .gitlab-ci.yml file for constructing pipelines
- Use container to run the jobs in pipelines for scalability

Our Technology Stack – Oracle automation



Pipeline and Runners



Our Gitlab Runner images

databases/databases-runner-images:2.14.8

General-purpose, Ansible playbook execution, gather information from target servers (test/control flow)

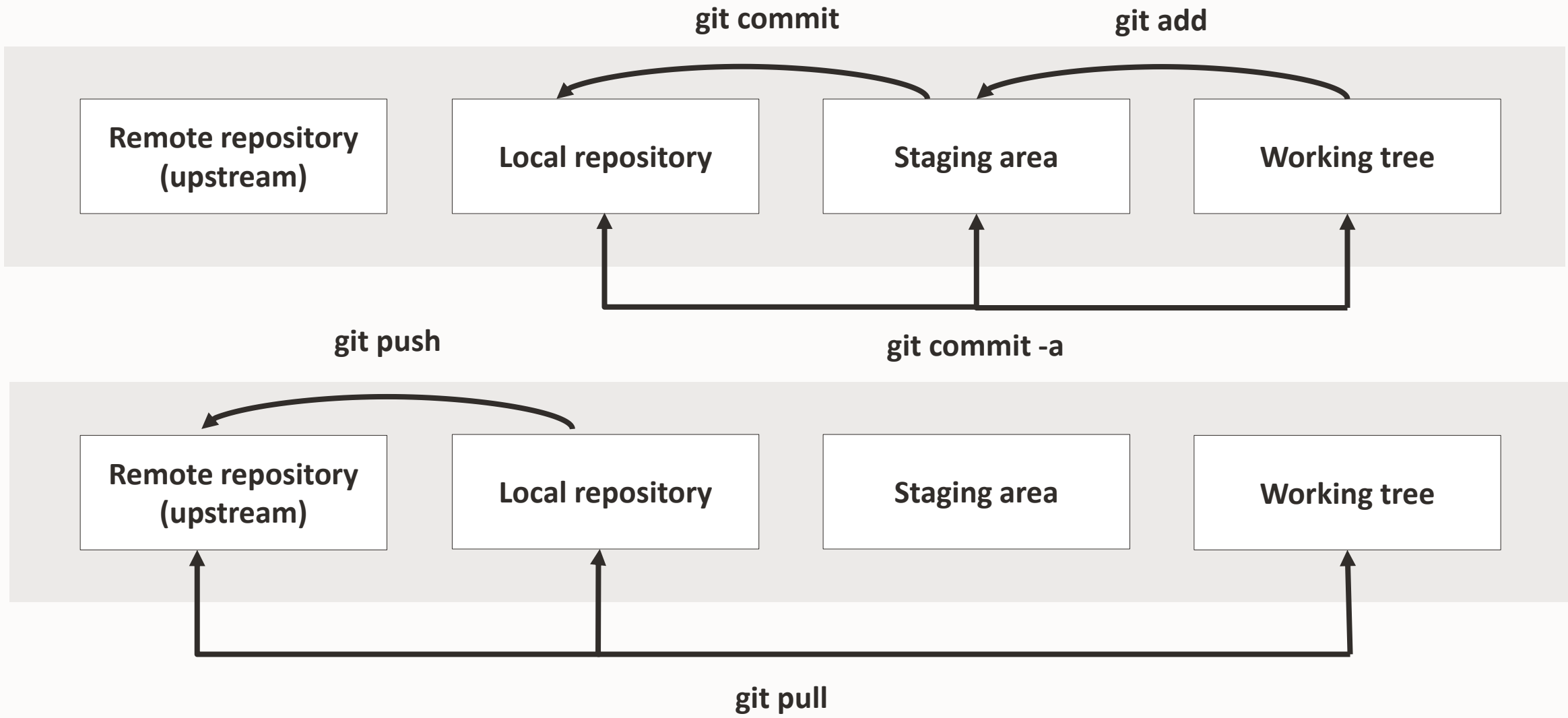
caas-runner-images/itsm-integration-runner:1.0.2

Validation (static + dynamic), Linting, CMDB maintenance,

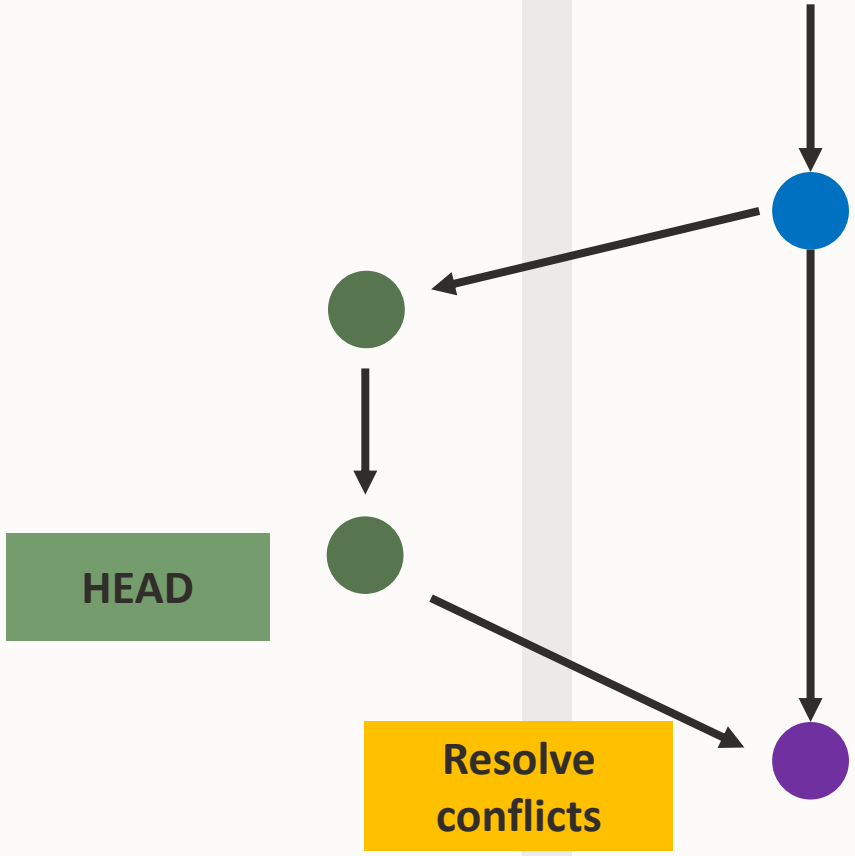
automated-testing/testcore-linux-image:BSE_1.1

Unit & Integration tests

Managing Material with Git – Primer (I)



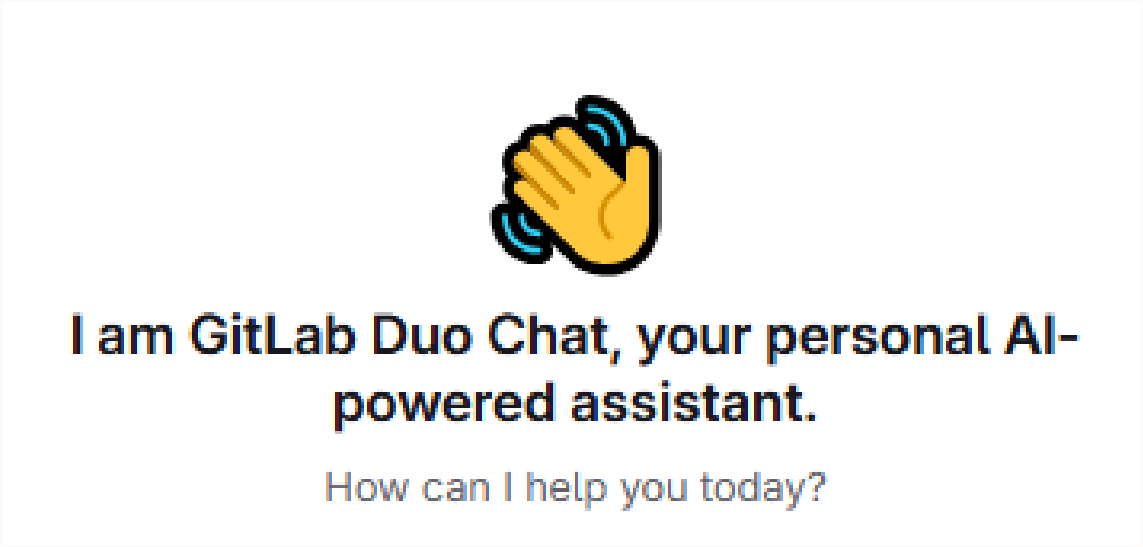
Managing Material with Git – Primer (II)



commit hash	commit message
82a59f93	
e91c50ab	Adding feature
761ea91	Updating dependencies
e018c8d9	Merging feature/a into main

Merge Request

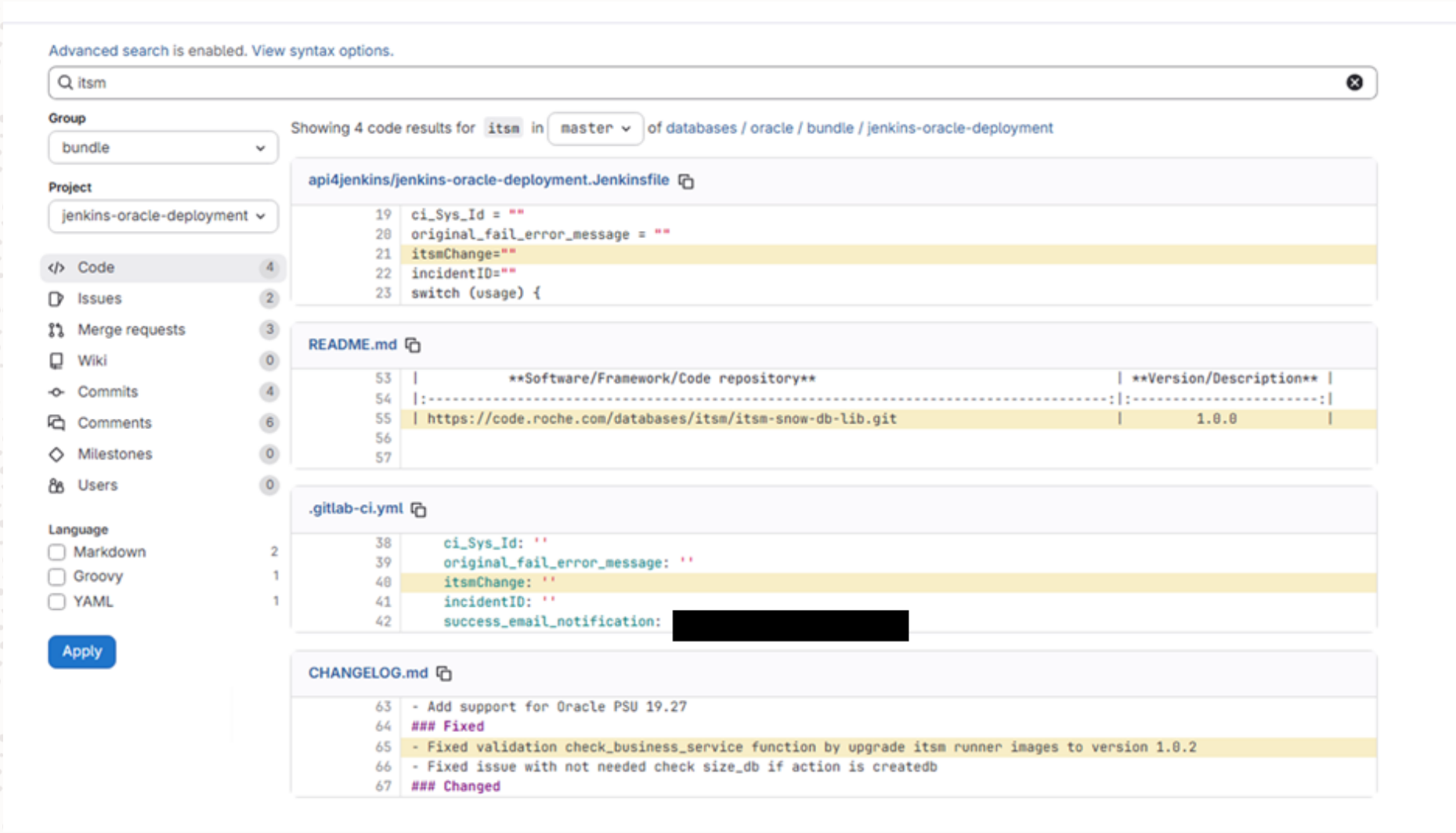
AI in Gitlab



Features

- Code generation
- Tests generation
- Code refactoring
- Chat
- Agent mode (in beta)
- Integration with Self-hosted LLM models

Don’t assume everything is in the toolbox already



„Recent Feature”

Oracle ACE Pro

Our Automation - summary

Ansible roles (63 roles)

supporting roles

- ansible-role-db-getserver-ci
- ansible-role-oracle-acl
- ansible-role-oracle-apex-add-workspace

leading roles

- ansible-role-oracle-db-deco
- ansible-role-oracle-cohesity-backup
- ansible-role-oracle_db_create
- ansible-role-oracle-autopsu-apply
- ansible-role-oracle-db-clone-pit
- ansible-role-oracle_soft_clone
- ansible-role-oracle-db-standby

Ansible orchestration (20 pipelines/playbooks)

- ansible-playbook-db-clone-pit
- ansible-playbook_oracle_db_deco
- ansible-playbook_oracle_db_autopsu
- ansible-playbook_oracle_db_deco_auto
- ansible-playbook_oracle_db_standby
- ansible-playbook_oracle_soft_install

Regression tests (1 multipipeline pipeline)

Example - Oracle IaaS Offering

Overview

Intended Use

What you Get

Highlights

INFRASTRUCTURE as a Service (IaaS)

Advanced Business Requirements
Sandpit, Dev, Test, Production

Oracle Instance

- Self-provision
- Gitlab platform used
- Self-managed operations
- Integrated with company core systems (SNOW, Cohesity, LogicMonitor)
- Oracle Cloud Control for management

Oracle IaaS Deployment
Provisioning

What we provide?

- ✓ ServiceNow CMDB updated with new Configuration Item and all upstream and downstream relations
- ✓ Validation of input parameters (static & dynamic)
- ✓ Backup automatically added for database (Cohesity/NAS) with proper retentions
- ✓ ITIL procedures automatically handled (Change Requests + Incidents)
- ✓ Email communication for deployment
- ✓ Automatic test cases performed during deployment
- ✓ Integration with RDBI management system
- ✓ Triggered by Automation Form or curl command (CLI mode)
- ✓ Integration with Mercury API for cli mode
- ✓ Integration with monitoring system - LogicMonitor

Pipeline

Actions

prepare

- Configure OS settings for Oracle database
- Create required file systems for Oracle database – datafiles and fra area
- Create oracle user with required profile
- Prepare environment for using automation Ansible playbooks and roles
- Best practice: do not create prerequisities manually. It could ends with situation when automatic tests fails and automatic cannot be used for some steps

installsw

- Install Oracle Software with desired version
- Best practice: avoid install Oracle Software with running database on servers. Timeout for operation is very likely.

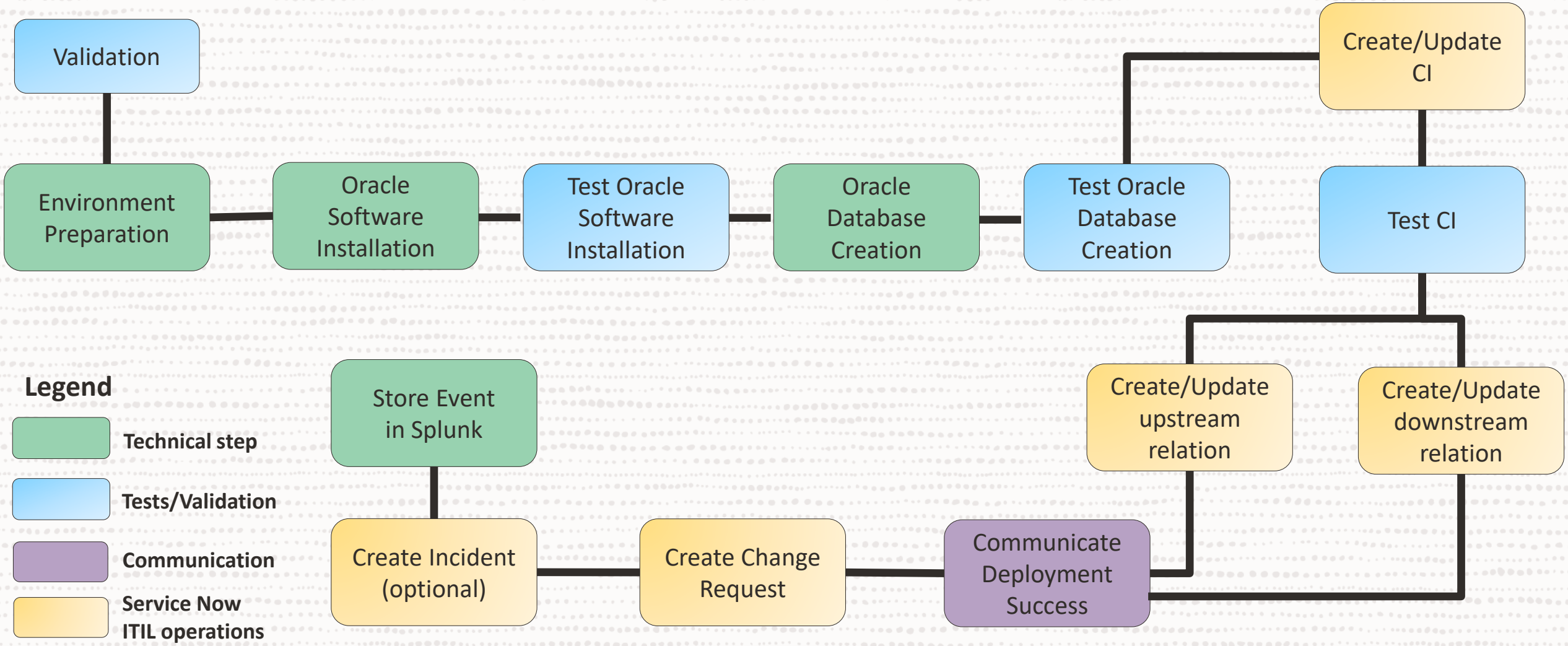
full

- = prepare + installsw + full

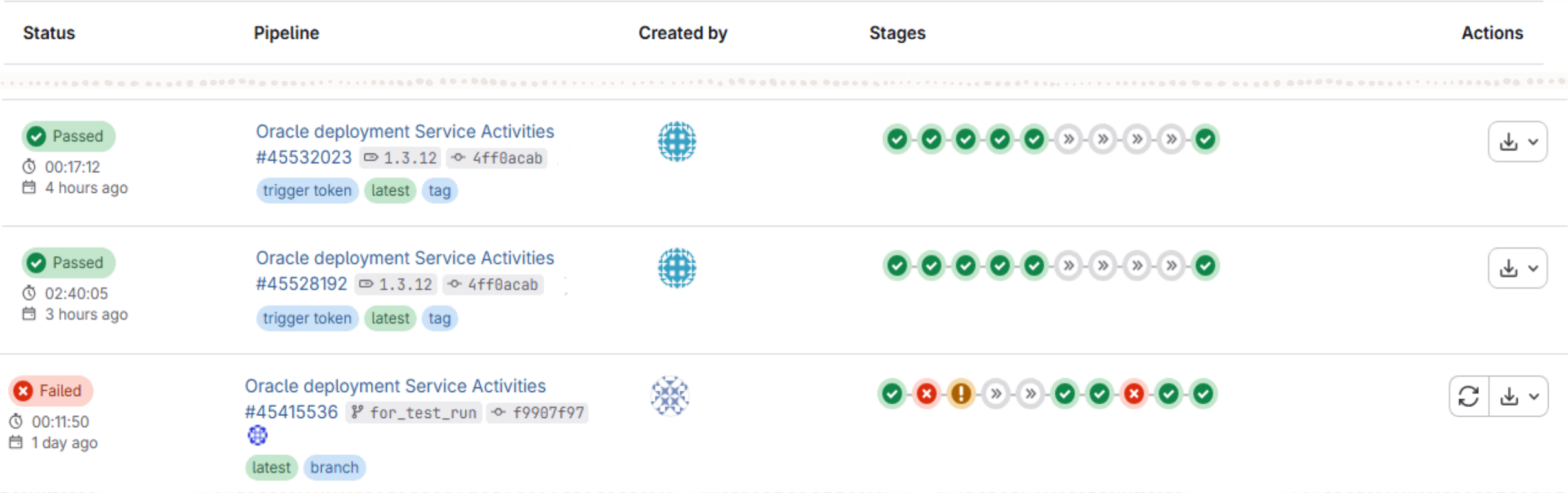
createdb

- Create and configure new database
- Create new database listener on dedicated port
- Create dedicated database profiles (password verification function and aging policies)
- Configure Cohesity/NAS backups with retention policies
- Best Practices: Capacity is controlled. When limit is reached for #databases new will not be created

Oracle Deployment Pipeline Diagram



Oracle Deployment Pipeline Executions



Oracle ACE Pro

Oracle Deployment Pipeline - Analytics

Commit statistics for master Feb 28 - Feb 20

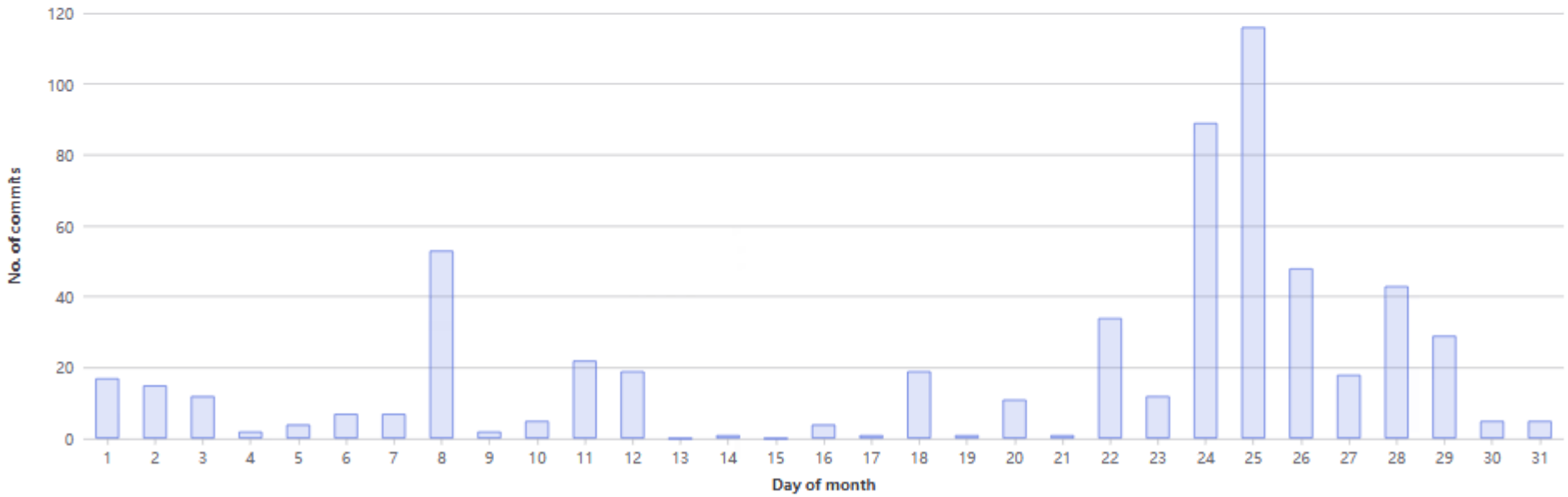
Excluding merge commits. Limited to 2,000 commits.

master

jenkins-oracle-deployment

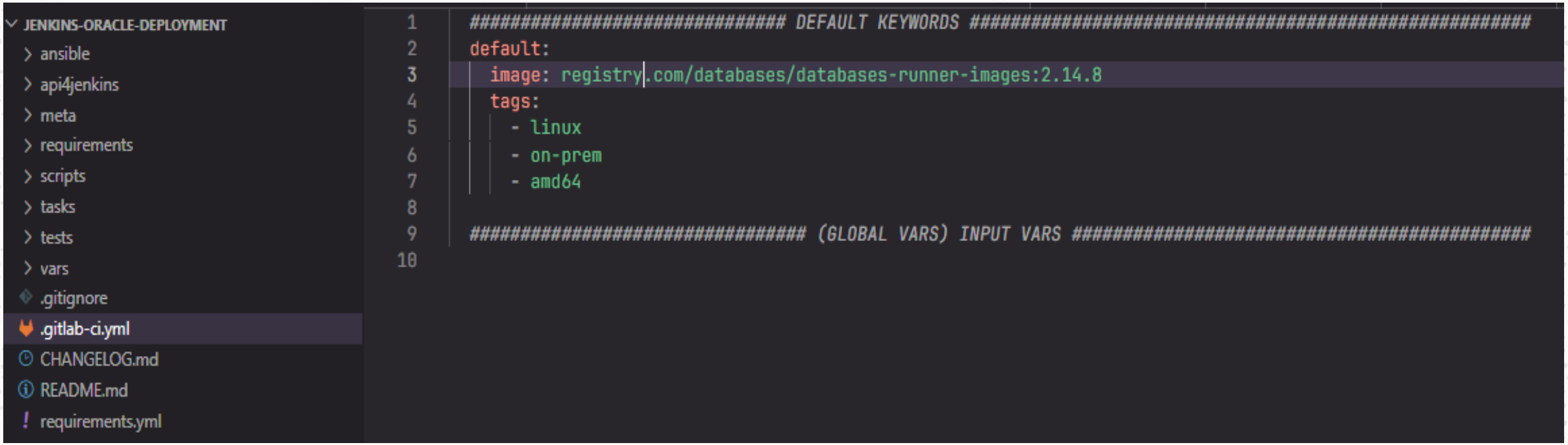
- Total: **602 commits**
- Average per day: **0.8 commits**
- Authors: **7**

Commits per day of month



Oracle ACE Pro

Gitlab Pipelines – code tour



Default Runner image for stages/jobs in Pipeline

Gitlab Pipelines – code tour

```
141  stages:
142  - validate_input
143  - launch_oracle_environment_preparation
144  - test_environment_preparation
145  - oracle_software_install
146  - test_oracle_soft_install
147  - oracle_db_create
148  - test_oracle_db_create
149  - oracle_db_create_update_ci
150  - test_ci_create_update
151  - oracle_create_ci_relations
152  - change_create_update_close_successful
153  - oracle_email_notification
154  - validate_execution_error_mesg
155  - change_create_update_close_failure
156  - create_incident
157  - oracle_email_notification_failure
158  - get_artifacts
```

Pipeline stages

Gitlab Pipelines – code tour

```yaml
test_environment_preparation:
  stage: test_environment_preparation
  image: registry.com/calm/automated-testing/testcore-linux-image:BSE_1.1
  script:
    - set -x
    - echo "$server ansible_ssh_private_key_file=id_rsa ansible_user=oracle ansible_ssh_extra_args='-o StrictHostKeyChecking=no'" >> inventory
    - echo "${database_key}" > ./id_rsa
    - chmod 400 ./id_rsa
    - pytest --hosts=$server  --force-ansible --hosts='ansible://all' --ansible-inventory=inventory --color=yes -v -W ignore::pytest.PytestExperimentalApiWarning --db_name $db_name
    - testsrc=$?
    - allure generate tests/reports
    - echo error_env_preparation=$error_env_preparation >> build.env
    - echo fs_extended=$fs_extended >> build.env
    - echo linux_env_preparation=$linux_env_preparation >> build.env
    - if [ $error_env_preparation == "true" ] || [ $fs_extended == "true" ] || [ $linux_env_preparation == "true" ] || [ $testsrc -gt 0 ]; then
    -     exit 1
    - fi
  rules:
    - if: $action == "prepare" || $action == "full"
  artifacts:
    when: always
    paths:
    - tests/reports
    - allure-report
    reports:
      junit: tests/reports/*.xml
      dotenv: build.env
  dependencies:
    - 'launch_oracle_environment_preparation'
    - 'Validate'
  when: always
  allow_failure: true
```

Test preparation stage (server + configuration)

Gitlab Pipelines – code tour

```python
import pytest

@pytest.mark.parametrize(
    "sysctl_key,expected,os8_valid,os9_valid",
    [
        ("fs.file-max", "6815744", "true", "true"),
        ("kernel.sem", "300\t128000\t200\t8192", "true", "true"),
        ("kernel.shmmni", "4096", "true", "true"),
        ("kernel.msgmni", "2878", "true", "true"),
        ("kernel.msgmax", "8192", "true", "true"),
        ("kernel.msgmnb", "65536", "true", "true"),
        ("kernel.sched_min_granularity_ns", "10000000", "true", "false"),
        ("kernel.sched_wakeup_granularity_ns", "15000000", "true", "false"),
        ("net.core.rmem_default", "262144",  "true", "true"),
        ("net.core.rmem_max", "4194304", "true", "true"),
        ("net.core.wmem_default", "262144", "true", "true"),
        ("net.core.wmem_max", "1048576", "true", "true"),
        ("fs.aio-max-nr", "3145728", "true", "true"),
        ("net.ipv4.ip_local_port_range", "9000\t65500", "true", "true"),
        ("vm.swappiness", "0", "true", "true"),
        ("vm.dirty_writeback_centisecs", "100", "true", "true"),
        ("vm.dirty_expire_centisecs", "500", "true", "true"),
        ("vm.dirty_background_ratio", "3", "true", "true"),
        ("vm.dirty_ratio", "15", "true", "true"),
    ]
)
@pytest.mark.test_prepare('test_environment_preparation')
def test_kernel_parameters(host, stage, os_version, record_xml_attribute, sysctl_key, expected, os8_valid, os9_valid):
```

Example preparation stage test

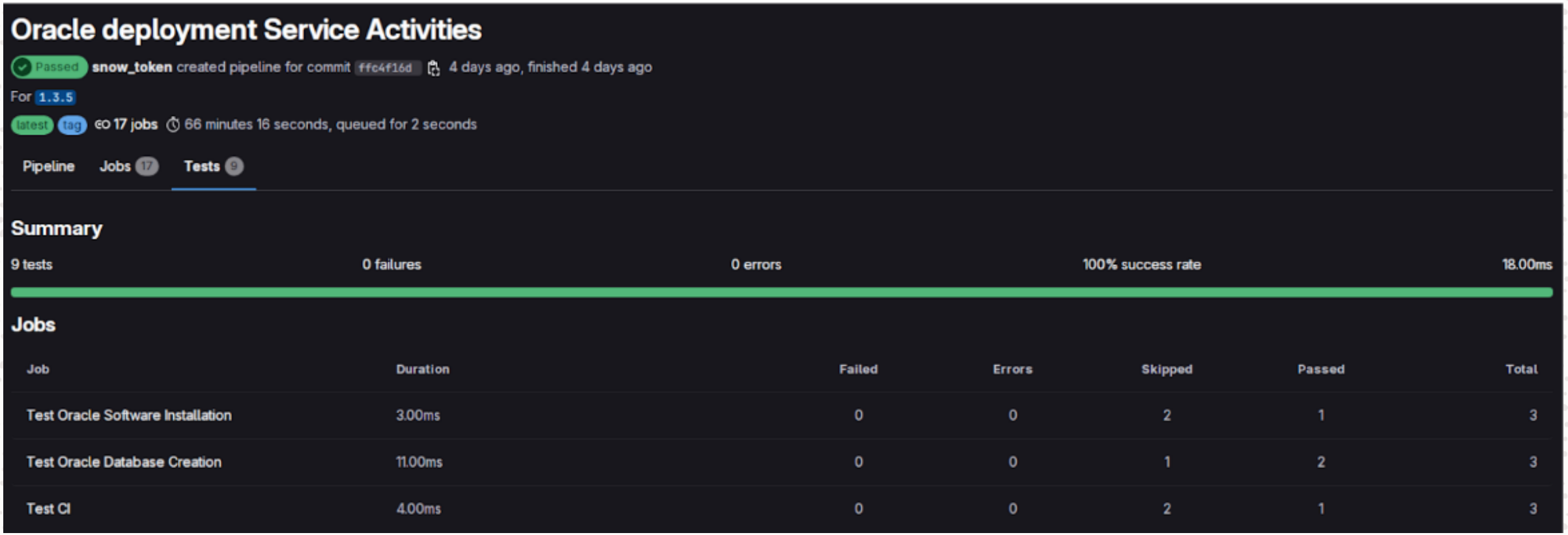
Gitlab Pipelines – code tour

```python
@pytest.mark.test_prepare('test_environment_preparation')
def test_kernel_parameters(host, stage, os_version, record_xml_attribute, sysctl_key, expected, os8_valid, os9_valid):

    record_xml_attribute("classname", stage)
    record_xml_attribute("name", "req007-kernel-parameters")
    run_test = False

    # evaluate if check should be run
    if os_version == "8" and os8_valid == "true":
        run_test = True
    elif os_version == "9" and os9_valid == "true":
        run_test = True

    # test kernel parameters
    if run_test:
        cmd = f'sysctl -a 2>/dev/null | grep "{sysctl_key} =" | tr -d " " | cut -d= -f2'
        res_cmd = host.run(cmd)
        assert res_cmd.succeeded == True
        assert res_cmd.stdout.strip() == expected
    else:
        pytest.skip()
```

Example preparation stage test

Tests – Gitlab UI

Test report for every pipeline run.



Gitlab Pipelines – code tour

```
Create, update upstream relations:
  stage: oracle_create_ci_relations
  extends: .snow_create_update_relation_uppstream
  image: registry.com/caas/caas-runner-images/itsm-integration-runner:1.0.2
  variables:
    REL_TYPE: ${cmdb_relation_impactservice}
    SNOW_BUSINESS_SERVICE_CLASS_NAME: "cmdb_ci_service_auto"
    SNOW_BUSINESS_SERVICE: ${business_service}
  rules:
    - if: $action == "createdb" || $action == "full"
  retry: 2
  needs: ["Create or update CI", "Oracle Database Creation"]
```

Update Relations in CMDB (Snow) using SNOW library

Oracle Database creation

```
- block:
  - name: run dbca to create the database
    shell: |
      export ORACLE_HOME={{ oracle_home }}
      $ORACLE_HOME/bin/dbca -silent -ignorePrereqs -createDatabase \
      -gdbName {{ ora_dbname }} \
      -templateName New_Database.dbt \
      -characterSet {{ ora_charset }} \
      -dvConfiguration false \
      -datafileDestination /var/opt/oracle/{{ ora_dbname }}/databases \
      -redoLogFileSize 500 \
      -systemPassword {{ ora_dbname }}adm1n \
      -sysPassword {{ ora_dbname }}adm1n \
      -olsConfiguration false \
      -createAsContainerDatabase false \
      -recoveryAreaDestination /var/opt/oracle/{{ ora_dbname }}/fra \
      -recoveryAreaSize {{ fra_mb }} \
      -createListener LISTENER_{{ ora_dbname }}:{{ port }} \
      -useOMF true \
      -totalMemory {{ ora_memory_mb }} \
      -dbOptions OMS:false,JSERVER:false,SPATIAL:false,IMEDIA:false,ORACLE_TEXT:false,CWMLITE:false,SAMPLE_SCHEMA
      -sampleSchema false \
      -databaseType MULTIPURPOSE \
      -initParams db_create_file_dest=/var/opt/oracle/{{ ora_dbname }}/databases,db_create_online_log_dest_1=/var
      -storageType FS \
      -databaseConfigType SINGLE \
      -emConfiguration NONE
    register: dbca_result
    failed_when: dbca_result.rc == 1
```

Example rollback scenario

```
270
271   - name: create file with the software installation outcome
272     copy:
273       dest: "{{ playbook_dir }}/oracle_soft_install_{{ version }}.out"
274       content: |
275         ORACLE DATABASE SOFTWARE SUCCESSFULLY INSTALLED
276     delegate_to: localhost
277
278   - debug:
279       msg: ORACLE DATABASE SOFTWARE SUCCESSFULLY INSTALLED
280
281   when: not skip ### conditional for the whole block
282
283   rescue:
284     - debug:
285         msg: ORACLE DATABASE SOFTWARE INSTALLATION FAILED! Executing rollback...
286
287     - name: restore backup of Oracle Inventory
288       unarchive:
289         src: "/opt/oracle/oraInventory_backup_{{ tstamp }}.tgz"
290         dest: /opt/oracle/oraInventory
291         remote_src: yes
292       when: orainv_exists
293
294     - name: remove oracle home directory
295       file:
296         path: "{{ oracle_home }}"
297         state: absent
298
299     - name: force playbook for failing and sending email
300       fail:
301         msg: "DATABASE SW CREATION FAILED! Failing Creation Sending Mail for check"
302
303   always:
304     - name: delete oracle software image
305       file:
306         path: "/opt/oracle/{{ sw_file }}"
307         state: absent
```

Tests (I)

Test Cases **Pytest library**

Unique ID	Test Name	Requirement	Acceptance Criteria
req001	test_deploy.py	Check Oracle Software / Oracle database have installed/created successfully	Oracle Software Installed
req002	test_qualifiedcheckbox.py	Test if Qualified checkbox is checked in SNOW for new database CI	Qualified checkbox checked in Configuration Item in Servicenow
req003	test_backup_retention.py	Test if backup retention for database has been set correctly	Backup retention is correct (prd – 21 days, nonprd – 14 days)

Tests (II)

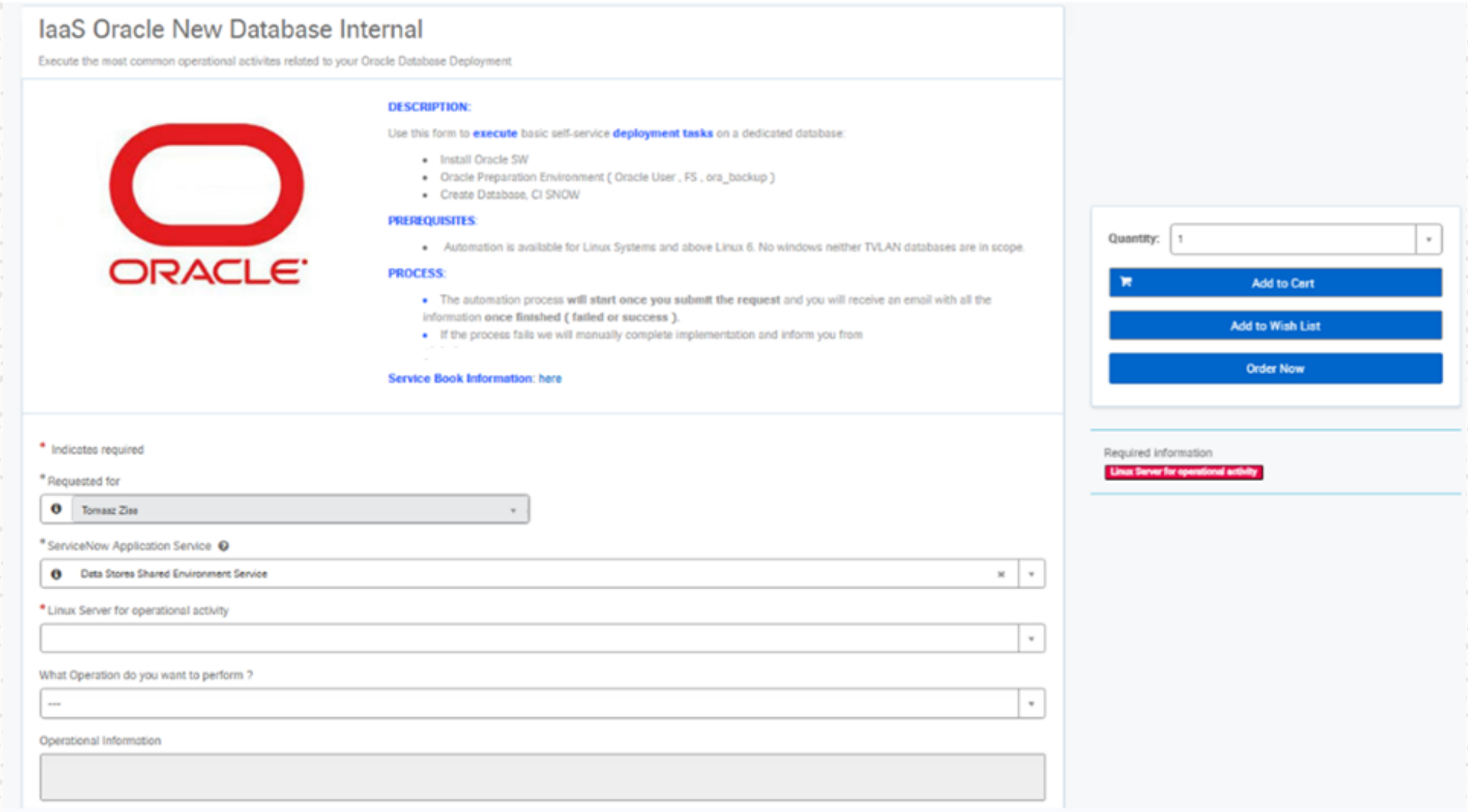
Pipeline Stages and Tests

Pipeline Stage	Fired test(s)	Not-fired test(s)
Test Oracle Software Installation	test_deploy.py	test_qualifiedcheckbox.py test_backup_retention.py
Test Oracle Database Creation	test_deploy.py test_backup_retention.py	test_qualifiedcheckbox.py
Test CI	test_qualifiedcheckbox.py	test_deploy.py test_backup_retention.py

Validation

Input Parameter Name	Check
business_service	Is not empty string If not called by Web Form then check if exists in ServiceNow
server	Is not empty string If not called by Web Form then check if exists in ServiceNow
db_name	Is not empty string Database name has maximum 8 characters
size_db	Is not empty string Is integer number
level_support	Is not empty string If not called by Web Form then if has value in set (‘Production’, ‘Test’, ‘Development’, ‘Sandpit’)
version	Is not empty string If not called by iCare Form then if value in in supported version: 12.2 18.3 18.6 18.8 18.12 18.13 18.14 19.14 19.15 19.17 19.18 19.19 19.20 19.21 19.22 19.23 19.24 19.26

Web Form for Deployment (I)



Oracle ACE Pro

Automation via API

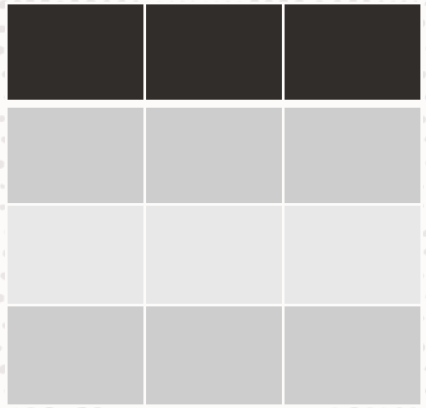
Pre-checks

```
curl -u USER:PASS -H 'Content-Type: application/json'
https://mercury.com/v1/form/oracle_deployment_operation/version/1/validate --data
'{"mail_address":„tomasz.ziss@example.com","userid":„tziss","request_id":"RITMIACAPI","ci_name":„PRD@host.com","
ora_check":"Y","operational_test":"N","oracle_version":"19.26","email_notification": "No"}'
```

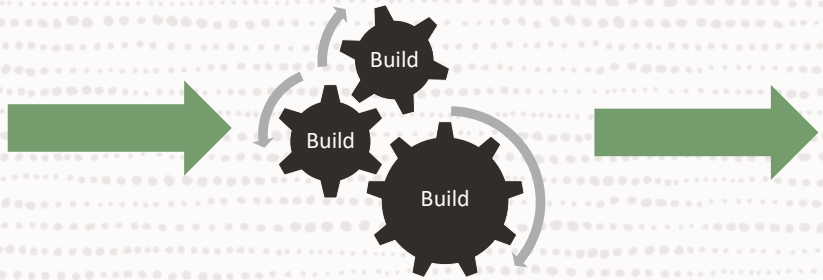
Submit

```
curl -u USER:PASS -H 'Content-Type: application/json'
https://mercury.com/v1/form/oracle_deployment_operation/version/1/submit --data
'{"version":"19.26","server":„example.com","action":"installsw","mail_address":„tomasz.ziss@example.com","request_id
":"RITMORAPRE","userid":" tziss"}'
```

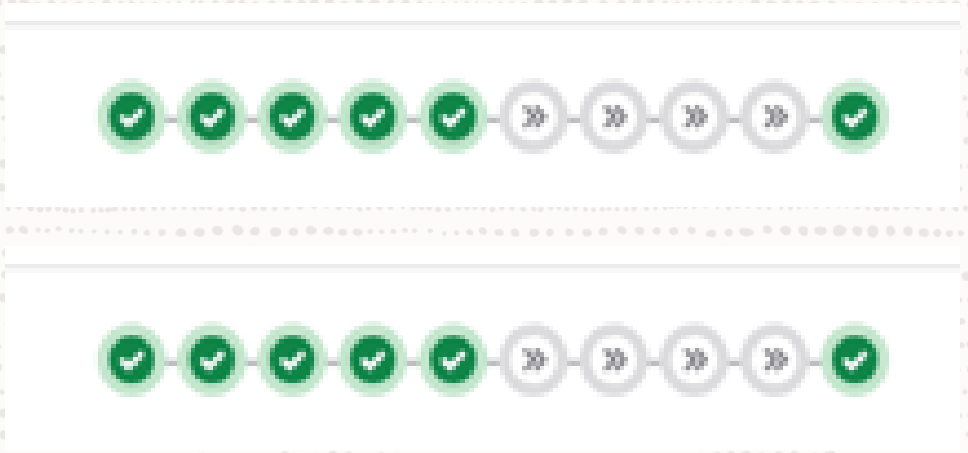
Automatic Regression tests (I)



Tests Sets
(Multipipeline defintion)



Build dynamically pipeline (multiple stages) for each Test Set

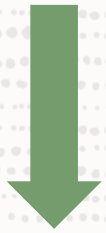


Pipelines for each TestSet

Dynamic child pipeline generation (YAML)

Automatic Regression tests (II)

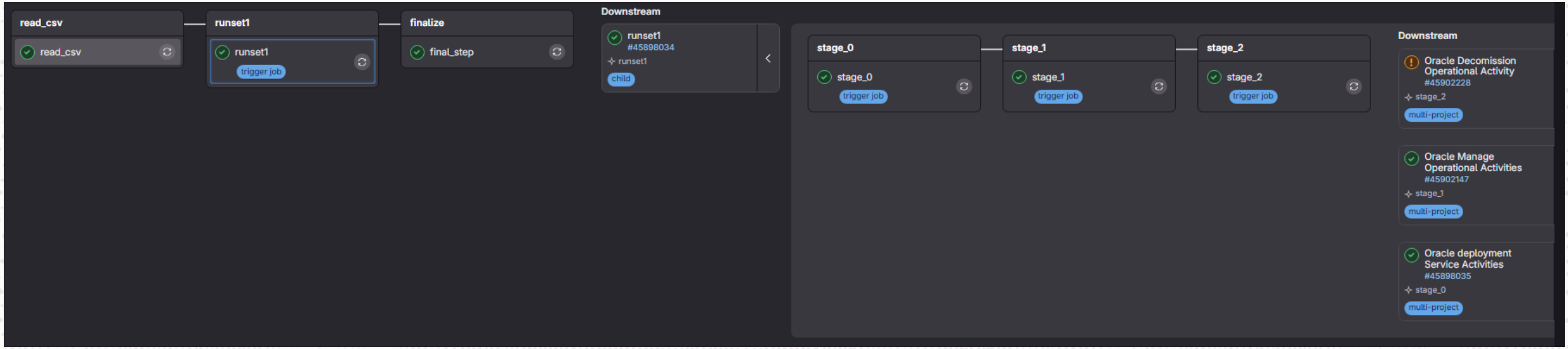
```
Set1,databases/oracle/bundle/jenkins-oracle-deployment,for_run,server:'servera.example.com',db_name:'T5NOP',size_db:'50',version:'19.29',
mail_address:'tomasz.ziss@example.com',action:'full'
Set1,databases/oracle/orchestrate/ansible-playbook_oracle_db_manage,for_run,operation:'USER',userid:'zisst',ci_name:'T5NOP@servera.example.
com',mail_address:'tomasz.ziss@example.com',schemas:'ZISST'
Set1,databases/oracle/orchestrate/ansible-playbook_oracle_db_deco,for_run,ci_name:'T5NOP@servera.example.com',mail_address:'tomasz.
ziss@example.com'
```



```yaml
read_csv:
  stage: read_csv
  image: registry.example.com/databases/databases-runner-images:2.14.8
  script:
    - pip install pandas
    - python scripts/generateSet1.py
  artifacts:
    paths:
      - pipeline_set1.yml
```

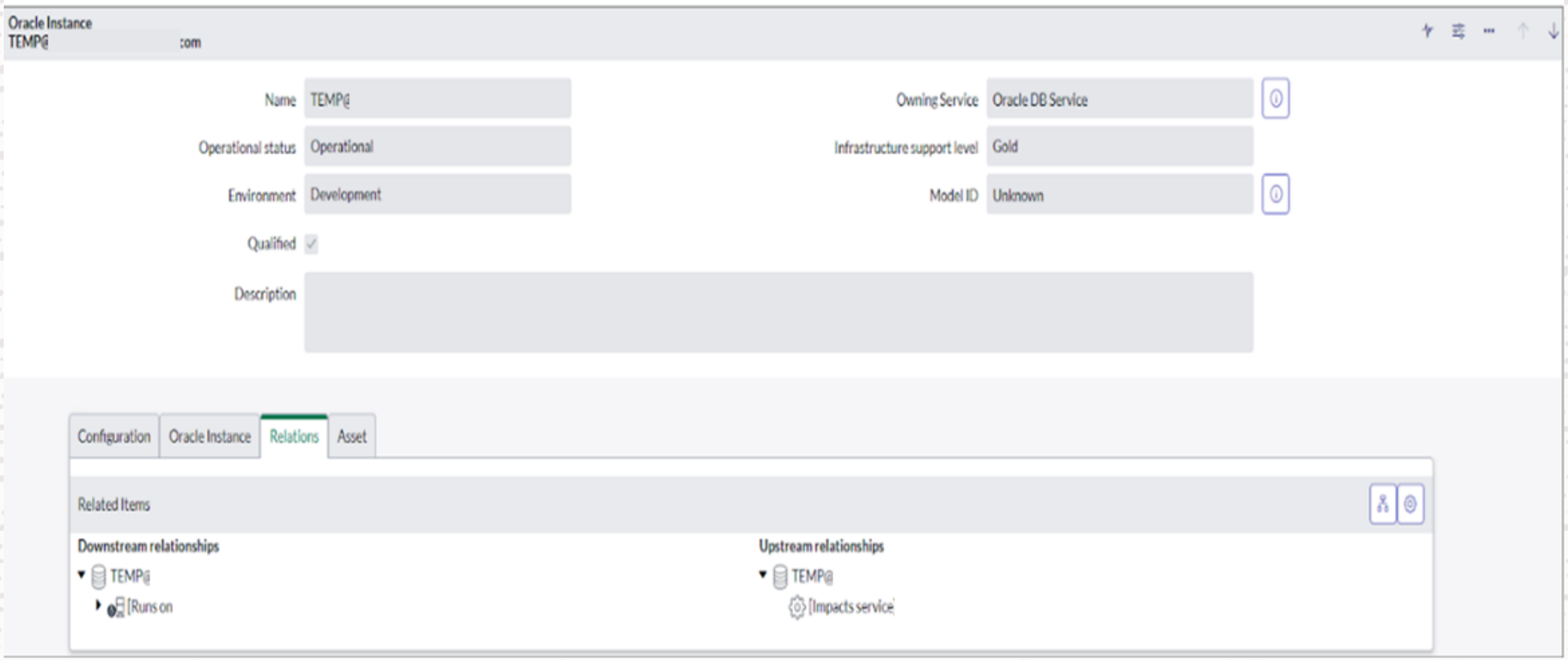
```yaml
stages: [stage_0, stage_1, stage_2]
stage_0:
  stage: stage_0
  trigger:
    project: databases/oracle/bundle/jenkins-oracle-deployment
    branch: for_run
    strategy: depend
  variables:
    server: 'serverA.example.com'
    db_name: 'T5NOP'
    size_db: '50'
    version: '19.29'
    mail_address: 'tomasz.ziss@example.com'
    action: 'full'
stage_1:
  stage: stage_1
  trigger:
    project: databases/oracle/orchestrate/ansible-playbook_oracle_db_manage
    branch: for_run
    strategy: depend
  variables:
    operation: 'USER'
    userid: 'zisst'
    ci_name: 'T5NOP@serverA.example.com'
    mail_address: 'tomasz.ziss@example.com'
    schemas: 'ZISST'
stage_2:
  stage: stage_2
  trigger:
    project: databases/oracle/orchestrate/ansible-playbook_oracle_db_deco
    branch: for_run
    strategy: depend
  variables:
    ci_name: 'tomasz.ziss@example.com'
    mail_address: 'tomasz.ziss@example.com'
```

Automatic Regression tests (III)



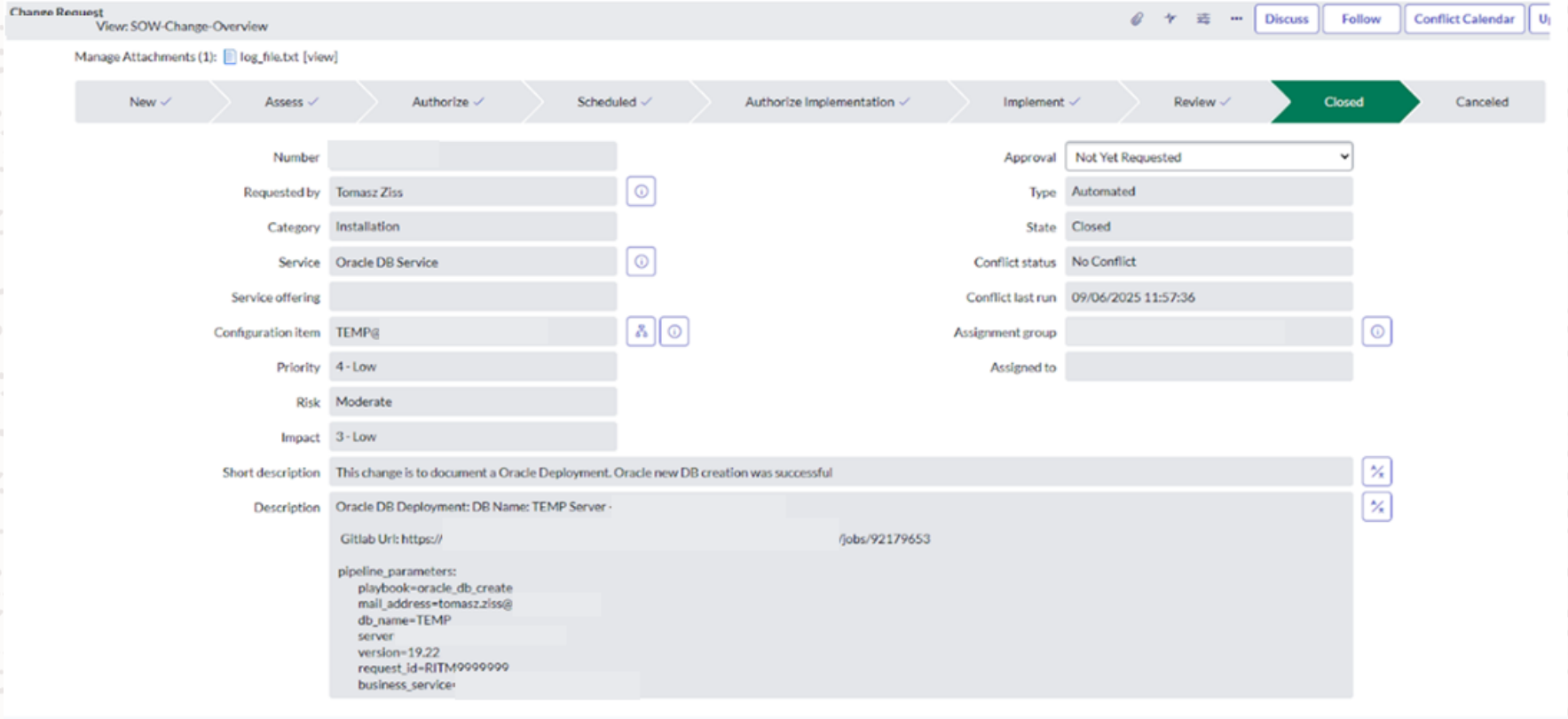
Oracle ACE Pro

Example Oracle Instance (CI) in SNOW



Oracle ACE Pro

Example Change Request in SNOW



Oracle ACE Pro

Example Incident in SNOW

Number	INC14657674	State	Resolved
Caller		Contact type	Webservice
Affected user		Impact	3 - Low
Additional contact		Urgency	3 - Low
Service offering	Oracle DB - Bronze	Priority	5 - Planning
Service	Oracle DB Service	Assignment group	
Business criticality	Critical		
Category	Infrastructure	Assigned to	
Subcategory	Other	Master Incident	
Configuration item			
Short description	Oracle Deployment Pipeline failed.		
Description	Incident created on Oracle deployment pipeline failure. Contact or more information Request RITM5303881		

Lessons Learnt

- Ensure pipeline with default parameters works correctly and perform enough tests as part of it.
- Remember that writing pipeline is not classic programming. Do not assume every programming tricks/techniques work exactly the same with as in programming language. Check code frequently.
- Design you artificats flow between stage in pipelines. If stage has more than 10 inputs from other stages it could be unstable
- Review your rollback and job retry strategy
- Gitlab – there is so many diffirent configurations (plugins/extentsions) across enterprises. Think portability and stability.
- Gitlab executor – are you big enough to use Kubernetes Executor?
- Use automatic test regression and TDD techniques – DevOps helps to perform a lot of automation executions.

Thank You ☺