Wrocław University of Science and Technology

Large Scale Data Processing

Lecture 11 – Automatic app management in the production

dr hab. inż. Tomasz Kajdanowicz, Piotr Bielak, Roman Bartusiak, Krzysztof Rajda

January 3, 2022

HR EXCELLENCE IN RESEARCH

Overview

Continous X

Data Schema updates

- Liquibase
- FlywayDB
- Alembic
- db-migrate

Helm

Ansible

Continuous Deployment (1)

Continous X

- What if continuous integration is not enough for you?
- What if you want to deploy your software as fast as possible after feature development?
- How to perform multiple deployments daily? (e.g. Facebook performs thousands of deployments per day)
- You need to be sure that your production environment is as fresh as it can be?

Continuous Deployment (2)

Continous X

- Deploy your software as part of your build pipeline (to production or production-like environment)
- Create push-button deployment possibilities
- Automate as much as you can
- CI do not guarantee that your software will be deploy-able, CD does

Continuous Deployment (3)

Continous X

- Create continuous integration pipeline with all its requirements
- Extend it to contain executable creation
- Create automation procedure that allows you to deploy your app/services
- Utilize appropriate updates procedure (canary etc.)
- Add deployment as part of build pipeline, or create special build for that

Continuous Deployment - Argo

Continous X

- Argo CD is a declarative, GitOps continuous delivery tool for Kubernetes.
- helm, kustomize, etc.
- plain k8s definitions
- UI
- pull and push
- declarative

Continuous Deployment - Flux

Continous X

- Flux is a set of continuous and progressive delivery solutions for Kubernetes that are open and extensible.
- helm, kustomize, etc.
- plain k8s definitions
- No UI
- pull and push
- declarative
- CNCF

Overview

Continous X

Data Schema updates

- Liquibase
- FlywayDB
- Alembic
- db-migrate

Helm

Ansible

Liquibase

Data Schema updates

- YAML, SQL, XML, JSON
- Java
- explicit

Liquibase

Data Schema updates

```
databaseChangeLog:
  - include:
      file: liquibase/01-create-sample-schema.yaml
```

Liquibase

Data Schema updates

```yaml
databaseChangeLog:
  - changeSet:
      id: 1
      author: Roman Bartusiak
      comment: "Sample changelog"
      changes:
        - createTable:
            tableName: sample
            columns:
              - column:
                  name: id
                  type: uuid
                  constraints:
                    primaryKey: true
                    nullable: false
```

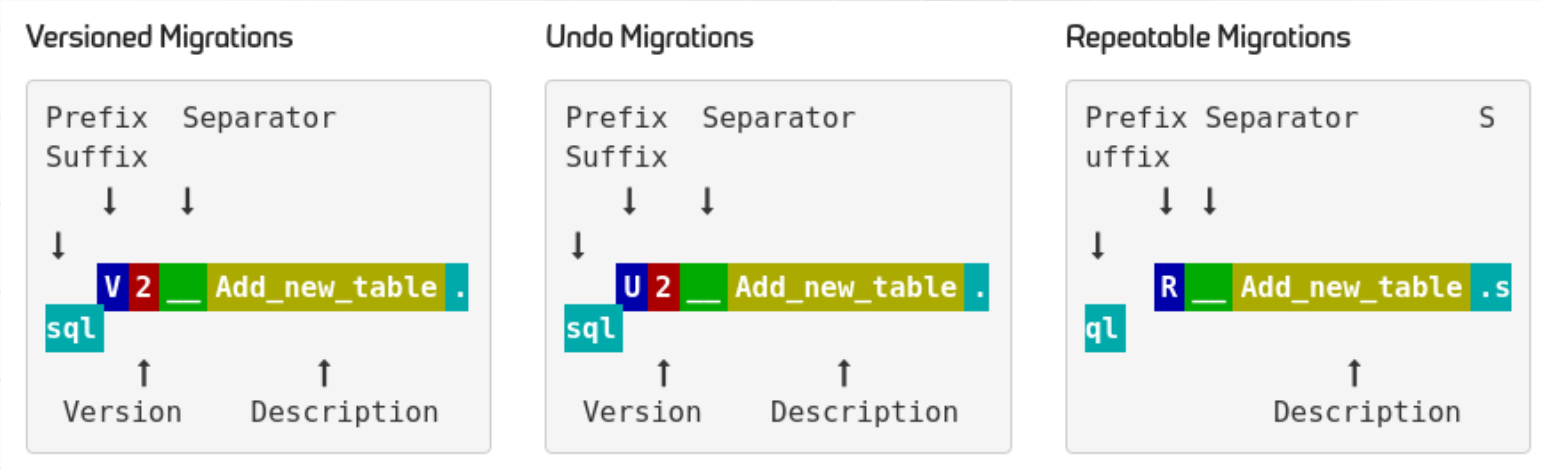
FlywayDB

Data Schema updates

- SQL
- Java
- file name based

FlywayDB

Data Schema updates



Alembic

Data Schema updates

- Python
- SQLAlchemy
- auto-generation

Alembic

Data Schema updates

```
1  """empty message
2
3  Revision ID: 27c6a30d7c24
4  Revises: None
5  Create Date: 2011-11-08 11:40:27.089406
6
7  """
8
9  # revision identifiers, used by Alembic.
10 revision = '27c6a30d7c24'
11 down_revision = None
12
13 from alembic import op
14 import sqlalchemy as sa
```

Alembic

Data Schema updates

```
1  def upgrade():
2      ### commands auto generated by Alembic - please adjust! ###
3      op.create_table(
4      'account',
5      sa.Column('id', sa.Integer()),
6      sa.Column('name', sa.String(length=50), nullable=False),
7      sa.Column('description', sa.VARCHAR(200)),
8      sa.Column('last_transaction_date', sa.DateTime()),
9      sa.PrimaryKeyConstraint('id')
10     )
11     ### end Alembic commands ###
12
13 def downgrade():
14     ### commands auto generated by Alembic - please adjust! ###
15     op.drop_table("account")
16     ### end Alembic commands ###
```

db-migrate

Data Schema updates

- Node.js
- scaffold generator

Wrocław University of Science and Technology

db-migrate

Data Schema updates

```
1  /* Promise-based version */
2  exports.up = function (db) {
3    return db.createTable('pets', {
4      id: { type: 'int', primaryKey: true },
5      name: 'string'
6    });
7  };
8
9  exports.down = function (db) {
10   return db.dropTable('pets');
11 };
```

Overview

Continous X

Data Schema updates

- Liquibase
- FlywayDB
- Alembic
- db-migrate

Helm

Ansible

Helm

- How to manage Kubernates deployments?
- There are not variables in Kubernates manifests
- What to do if we want to rollback to previous version

Helm

- Package manager for K8S
- OpenSource, maintained by Google, Microsoft, Bitnami, ...
- Many ready to use charts in official repository
- Possibility to use variables during deployment
- Charts can have dependencies

Helm

- Easily find packages
- Easily create new packages
- Can operate on any K8S cluster
- Query the cluster to see installed packages
- Update, delete, rollback and check history of installed charts

Helm

- Basically it templates K8S manifests (Go templates)
- Thanks to that, it is independent from resources that you want to create
- Create resource YAML, utilize {{ }} to inject variables
- Variables can be provided as a file or during the deployment

Helm

Helm v2

- requires Tiler
  - server-side (on cluster) component
  - manages packages
- 2-way strategic merge path
- releases names are global
- release name is optional, if not provided will be generated

Helm v3

- no Tiler
- 3-way strategic merge path
- releases names are in namespaces
- release name is required
- library charts

Directory structure

Helm (db-migrate)

```
wordpress/
  Chart.yaml          # A YAML file containing information
                      #  about the chart
  LICENSE             # OPTIONAL: A plain text file containing
                      #  the license for the chart
  README.md           # OPTIONAL: A human-readable README file
  values.yaml         # The default configuration values
                      #  for this chart
  values.schema.json  # OPTIONAL: A JSON Schema for imposing
                      #  a structure on the values.yaml file
  charts/             # A directory containing any charts upon
                      #  which this chart depends.
  crds/               # Custom Resource Definitions
  templates/          # A directory of templates that,
                      #  when combined with values, will
                      #  generate valid Kubernetes manifest files.
  templates/NOTES.txt # OPTIONAL: A plain text file containing
                      #  short usage notes
```

Example template

Helm

```
1  apiVersion: v1
2  kind: ConfigMap
3  metadata:
4    name: {{ .Release.Name }}-configmap
5  data:
6    myvalue: "Hello World"
```

Example template

Helm

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
data:
  myvalue: "Hello World"
  drink: {{ .Values.favoriteDrink }}
```

```
helm install --set favoriteDrink=monsterek ./mychart
```

Helm

- How to check if your template will work?
- dry-run will render template to STD
- you need to check it manually
- rendered template can be not accepted by kubernetes

Overview

Continous X

Data Schema updates

- Liquibase
- FlywayDB
- Alembic
- db-migrate

Helm

Ansible

Introduction

Ansible

- open source automation tool,
- machine provision,
- configuration management,
- application deployment,
- YAML-based declarative language,
- agentless (uses SSH / Powershell),

Characteristics

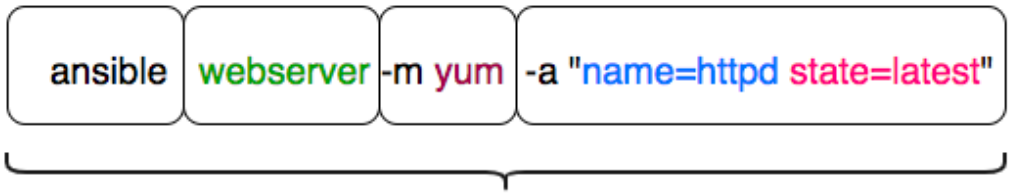
Ansible

- simple & minimalistic (YAML-based language & Jinja templates),
- consistency (of created enviroments),
- security (no dedicated agent, SSH used for connections),
- reliability (**idempotent** playbooks),

Command vs playbook

Ansible (db-migrate)

AD HOC command



Ansible Playbook

```
---
- name: playbook name
  hosts: webserver
  tasks:
    - name: name of the task
      yum:
        name: httpd
        state: latest
```

www.middlewareinventory.com

Example playbook

Ansible

```
1  ---
2  - name: Install nginx
3    hosts: all
4    become: true
5
6    tasks:
7    - name: Add epel-release repo
8      yum:
9        name: epel-release
10       state: present
11
12   - name: Install nginx
13     yum:
14       name: nginx
15       state: present
16
17   - name: Insert Index Page
18     template:
19       src: index.html
20       dest: /usr/share/nginx/html/index.html
21
22   - name: Start NGiNX
23     service:
24       name: nginx
25       state: started
```

Concepts (1)

Ansible

- playbooks:
  - define steps to build environments,
  - can be divided into multiple files (readability, reusability),
  - roles, vars, group_vars etc.
- modules:
  - define actual actions executed by Ansible,
  - examples given on previous slide,
  - standalone,
  - can be written in most scripting languages (Python, Bash, Perl, Ruby etc.),
  - should follow **idempotent** rule,

Concepts (2)

Ansible

- inventory file:
  - description of nodes that can be accessed by Ansible,
  - INI or YAML format,
  - IP addresses or hostnames,
  - when necessary SSH keys and users can be provided,
  - nodes can be assigned to groups,

Inventory file example

Ansible

```
1  # Consolidation of all groups
2  [hosts:children]
3  web-servers
4  offsite
5  onsite
6  backup-servers
7
8  [web-servers]
9  server1 ansible_host=192.168.0.1 ansible_port=1600
10 server2 ansible_host=192.168.0.2 ansible_port=1800
11
12 [offsite]
13 server3 ansible_host=10.160.40.1 ansible_port=22 ansible_user=root
14 192.168.6.1
15
16 # You can make groups of groups
17 [offsite:children]
18 backup-servers
19
20 [onsite]
21 server5 ansible_host=10.150.70.1 ansible_ssh_pass=password
22
23 [backup-servers]
24 foo.example.com
```

Next week

Ansible

- Języki do przetwarzania danych masowych

Wrocław University of Science and Technology

Large Scale Data Processing

Lecture 11 – Automatic app management in the production

dr hab. inż. Tomasz Kajdanowicz, Piotr Bielak, Roman Bartusiak, Krzysztof Rajda

January 3, 2022