AGENDA
Agenda

- Definition of YAML
- Why YAML?
- Relation to JSON and XML
- Preview of YAML
- Properties vs YAML

Usages of YAML

- Ansible Playbook
- Python – examples
- Kubernetes:
  - POD creation
  - Service creation
  - Deployment creation
  - Basic and Advanced yaml features

- Code Pattern
Definition of YAML

“YAML is a human friendly data serialization standard for all programming languages.”

- www.yaml.org
Why YAML?

- Easily readable by humans, expressive and extensible.
- Easy to implement and use.
- Easily portable between programming languages.
- Matches the **native data structures** of agile languages.
- Has a consistent model to support generic tools.
- Supports one-pass processing.

- Convenience: You’ll no longer have to add all of your parameters to the command line
- Maintenance: YAML files can be added to source control, so you can track changes
- Flexibility: You’ll be able to create much more complex structures using YAML than you can on the command line
COMPARISON BETWEEN XML, JSON AND YAML
XML vs JSON vs YAML

XML:
```xml
<Servers>
  <Server>
    <name>Server1</name>
    <owner>John</owner>
    <created>12232012</created>
    <status>active</status>
  </Server>
</Servers>
```

JSON:
```json
{  
  Servers: [  
    {  
      name: Server1,  
      owner: John,  
      created: 12232012,  
      status: active,  
    }  
  ]}
```

YAML:
```yaml
Servers:  
-  
  name: Server1  
  owner: John  
  created: 12232012  
  status: active
```
PREVIEW OF YAML
Key Value Pair, Array/Lists, Dictionary/Map

Key Value Pair
- Fruit: Apple
- Vegetable: Carrot
- Liquid: Water
- Meat: Chicken

Array/Lists
- Fruits:
  - Orange
  - Apple
  - Banana
- Vegetables:
  - Carrot
  - Cauliflower
  - Tomato

Dictionary/Map
- Banana:
  - Calories: 105
  - Fat: 0.4 g
  - Carbs: 27 g
- Grapes:
  - Calories: 62
  - Fat: 0.3 g
  - Carbs: 16 g
Importance of SPACES in YAML

Dictionary/Map

Banana:
  Calories: 105
  Fat: 0.4 g
  Carbs: 27 g
Importance of SPACES in YAML
NEVER use tabs in a YAML file
Complex YAML structures

Fruits:
- Banana:
  Calories: 105
  Fat: 0.4 g
  Carbs: 27 g

- Grape:
  Calories: 62
  Fat: 0.3 g
  Carbs: 16 g
Complex YAML structures

**Dictionary/Map**

- **Banana:**
  - Calories: 105
  - Fat: 0.4 g
  - Carbs: 27 g

- **Banana:**
  - Calories: 105
  - Carbs: 27 g
  - Fat: 0.4 g

**Array/List**

- **Fruits:**
  - Orange
  - Apple
  - Banana

- **Fruits:**
  - Orange
  - Banana
  - Apple
YAML BASICS IN A SLIDE
YAML Basics in one slide

---
!clarkevans.com/^invoice
invoice: 34843
date : 2001-01-23
bill-to: &id001
given : Chris
family : Dumars
address:
  lines: |
    458 Walkman Dr.
    Suite #292
city : Royal Oak
state : MI
postal : 48046
ship-to: *id001
product:
  - sku : BL394D
    quantity : 4
    description : Basketball
    price : 450.00
  - sku : BL4438H
    quantity : 1
    description : Super Hoop
    price : 2392.00
tax : 251.42
total: 4443.52
comments: >
  Late afternoon is best.
  Backup contact is Nancy Billsmer @ 338-4338.
YAML VALIDATOR
YAML TO JSON CONVERSION
YAML Validator/Formatter

- [https://jsonformatter.org/yaml-formatter](https://jsonformatter.org/yaml-formatter)
YAML Conversion to JSON

- [https://www.json2yaml.com/convert-yaml-to-json](https://www.json2yaml.com/convert-yaml-to-json)
- [https://yaml-online-parser.appspot.com/](https://yaml-online-parser.appspot.com/)

```
YAML

---
# <--- yaml supports comments, json does not
# did you know you can embed json in yaml?
# try uncommenting the next line
# { foo: 'bar' }

json:
  - rigid
  - better for data interchange

yaml:
  - slim and flexible
  - better for configuration

object:
  key: value
  array:
    - null_value:
    - boolean: true
    - integer: 1

paragraph: >
  Blank lines denote
  paragraph breaks

content: |
  Or we can auto
  convert line breaks
  to save space

JSON

{
  "json": [
    "rigid",
    "better for data interchange"
  ],
  "yaml": [
    "slim and flexible",
    "better for configuration"
  ],
  "object": {
    "key": "value",
    "array": [
      {
        "null_value": null
      },
      {
        "boolean": true
      },
      {
        "integer": 1
      }
    ]
  },
  "paragraph": "Blank lines denote\nparagraph breaks\n", "content": "Or we\ncan auto\nconvert line breaks\ninto save\nspace"
}
COMPARISON BETWEEN PROPERTIES FILE AND YAML
Properties file vs YAML

```properties
# application.properties
spring.datasource.url = jdbc:mysql://mysql-standalone:3306/test
spring.datasource.username = sa
spring.datasource.password = password

# Keep the connection alive if idle for a long time
spring.datasource.testWhileIdle = true
spring.datasource.validationQuery = SELECT 1

# Show or not log for each sql query
spring.jpa.show-sql = true

# Hibernate ddl auto (create, create-drop, update)
spring.jpa.hibernate.ddl-auto = update

# Naming strategy
spring.jpa.hibernate.naming-strategy = org.hibernate.cfg.Improved

# Use spring.jpa.properties.* for Hibernate
spring.jpa.properties.hibernate.type = org.hibernate.dialect.MySQL5Dialect

server.port=8086
```

```yaml
# application.yml
spring:
  datasource:
    url: jdbc:mysql://mysql-standalone:3306/test
    username: sa
    password: password
    testWhileIdle: true
    validationQuery: select 1
  jpa:
    show-sql: true
    hibernate:
      ddl-auto: update
      naming-strategy: org.hibernate.cfg.Improved
    properties:
      hibernate:
        dialect: org.hibernate.dialect.MySQL5Dialect
  server:
    port: 8086
```
YAML IN ANSIBLE
ANSIBLE PLAYBOOK

- Playbook – A single YAML file
  - Play – Defines a set of activities (tasks) to be run on hosts
    - Task – An action to be performed on the host
      - Execute a command
      - Run a script
      - Install a package
      - Shutdown/Restart
ANSIBLE PLAYBOOK FORMAT

#Simple Ansible Playbook1.yml

- name: Play 1
  hosts: localhost
  tasks:
  - name: Execute command ‘date’
    command: date
  - name: Execute script on server
    script: test_script.sh

- name: Play 2
  hosts: localhost
  tasks:
  - name: Install web service
    yum:
      name: httpd
      state: present
  - name: Start web server
    service:
      name: httpd
      state: started
YAML IN PYTHON
YAML in Python

```bash
pip3 install pyyaml
```
import requests
import json

url = 'http://192.168.2.1/rest/v3/
creds = {'user_name': 'admin', 'password': 'admin'}

s = requests.Session()
r = s.post(url + 'login-sessions', data=json.dumps(creds), timeout=1)
cookie_response = r.json()['cookie']
if r.status_code != 201:
    print('Login error, status code {0}'.format(r.status_code))

import requests
import json

url = 'http://192.168.2.1/rest/v3/
creds = {'user_name': 'admin', 'password': 'admin'}

s = requests.Session()
r = s.post(url + 'login-sessions', data=json.dumps(creds), timeout=1)
cookie_response = r.json()['cookie']
if r.status_code != 201:
    print('Login error, status code {0}'.format(r.status_code))
YAML in Python

```python
creds:
    userName: admin
    password: admin

import requests
import json
import yaml

def yaml_loader(filepath):
    with open(filepath, "r") as file_descriptor:
        data = yaml.load(file_descriptor)
    return data

filepath = "testexample.yaml"
data = yaml_loader(filepath)

s = requests.Session()
r = s.post(data['url'] + 'login-sessions', data=json.dumps(data['creds'])), timeout=1)
cookieResponse = r.json()['cookie']
if r.status_code != 201:
    print('Login error, status code {}'.format(r.status_code))
```
import yaml

def yaml_loader(filepath):
    with open(filepath, "r") as file_descriptor:
        data = yaml.load(file_descriptor)
    return data

def yaml_dump(filepath, data):
    with open(filepath, "w") as file_descriptor:
        yaml.dump(data, file_descriptor)

if __name__ == "__main__":
    filepath = "test.yaml"
    data = yaml_loader(filepath)
    print(data)

    filepath2 = "test2.yaml"
    data2 = {
        "items": {
            "sword": 100,
            "axe": 80,
            "boots": 40
        }
    }
    yaml_dump(filepath2, data2)
YAML in Python

Input: test.yaml

Output of YAML Load:
{'items': {'sword': 100, 'axe': 80}}

Output of YAML Dump:
items: {'axe': 80, 'boots': 40, 'sword': 100}
YAML IN KUBERNETES
Usage in Kubernetes: POD creation

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
labels:
  app: nginx
spec:
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
```
Usage in Kubernetes: Service creation

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service-4
labels:
  app: nginx
spec:
  type: NodePort
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
    nodePort: 30091
  selector:
    app: nginx
    tier: display
```
Usage in Kubernetes: Deployment creation

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginxdeploy-4
  labels:
    app: nginx
spec:
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: nginx
        tier: display
    spec:
      containers:
      - image: nginx
        name: front-end
        imagePullPolicy: Always
```
Kubernetes deployment: Basic specification

<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiVersion</td>
<td>Describes the versioned schema of the deployment object defined by the yaml specification file</td>
</tr>
<tr>
<td>Kind</td>
<td>Describes the resource defined by the yaml specification file. Kubernetes can infer kind based on the endpoint to which the specification is submitted</td>
</tr>
<tr>
<td>metadata.name</td>
<td>Deployments are uniquely identified within a namespace by their name.</td>
</tr>
<tr>
<td>metadata.namespace</td>
<td>Namespaces allow you to create separations within your cluster. Typically, objects can only communicate with other objects if they share a namespace. If the namespace field is omitted from your specification, the namespace ‘default’ is used</td>
</tr>
<tr>
<td>spec</td>
<td>The spec object is the real meat of the deployment specification yaml. It is where you define the actual desired behavior of the Kubernetes Deployment.</td>
</tr>
<tr>
<td>spec.template</td>
<td>Describes the Pods Kubernetes should create as part of the deployment.</td>
</tr>
<tr>
<td>spec.template.spec</td>
<td>Provides the explicit instructions for the desired behavior of a Pod</td>
</tr>
<tr>
<td>spec.template.spec.containers</td>
<td>An array of objects that must include at least 1 container. Each of the containers in this array describes the behavior of a containerized application within a Pod</td>
</tr>
<tr>
<td>spec.template.spec.containers[i].name</td>
<td>Each container within a Pod must have a unique name</td>
</tr>
<tr>
<td>spec.template.spec.containers[i].image</td>
<td>Describes which Docker image should run in the deployed container.</td>
</tr>
</tbody>
</table>
Kubernetes deployment: Advanced specification

How to control how many Pods are created in a deployment?

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  template:
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
```
Kubernetes deployment: Advanced specification

How to control the strategy with which Kubernetes replaces existing Pods?

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
    replicas: 3
  template:
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
```
Kubernetes deployment: Advanced specification

How to dictate when a Pod is available after an update?

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  minReadySeconds: 10
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  replicas: 3
  template:
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
```
Kubernetes deployment: Advanced specification

How to control how many revisions of a given deployment are kept by Kubernetes?

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  revisionHistoryLimit: 5
  minReadySeconds: 10
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  replicas: 3
  template:
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
```
Kubernetes deployment: Advanced specification

How to add labels to my Kubernetes deployment specification?

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  revisionHistoryLimit: 5
  minReadySeconds: 10
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
deployer: distelli
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
```
Kubernetes deployment: Advanced specification

Use label selectors in a deployment specification to tell my deployment to only target pods with certain labels?

```yaml
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  revisionHistoryLimit: 5
  minReadySeconds: 10
  selector:
    matchLabels:
      app: nginx
      deployer: distelli
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
    replicas: 3
  template:
    metadata:
      labels:
        app: nginx
        deployer: distelli
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
```
Developer First Framework

Usage of YAML: Code Pattern


• Spring Boot is an opinionated framework for quickly building production-ready Spring applications.

• This journey shows you how to create and deploy Spring Boot microservices within a polyglot application and then deploy the app to a Kubernetes cluster. It has heavy usage of YAML

Signup for IBM Cloud
https://bluemix.net

Stay Connected and continue coding!

Code & instructions
https://github.com/IBMDevConnect
https://github.com/IBM
https://github.com/IBM-Cloud
https://ibm-cloud.github.io/#/
http://ibm.github.io
https://github.com/watson-developer-cloud
https://github.com/ibm-bluemix-mobile-services

Apply for IBM Global Entrepreneur Program
https://developer.ibm.com/startups

Join our Meetup groups
Bangalore:
https://www.meetup.com/IBMDevConnect-Bangalore

Delhi / Gurugram / Noida:
https://www.meetup.com/ibmcloudcosyste

Mumbai / Pune:

Hyderabad / Vishakapatnam:
https://www.meetup.com/Hyderabad-Cognitive-with-Cloud

Recipes
https://developer.ibm.com/recipes/

Join our Slack team and stay in touch with the experts
https://ibmdevconnect.slack.com

Send in your request
http://ibm.biz/slackrequest
Thank you
GREAT INDIAN DEVELOPER SUMMIT 2019

Conference: April 23-26, Bangalore

Register early and get the best discounts!

www.developersummit.com  @greatindiandev  bit.ly/gidslinkedin