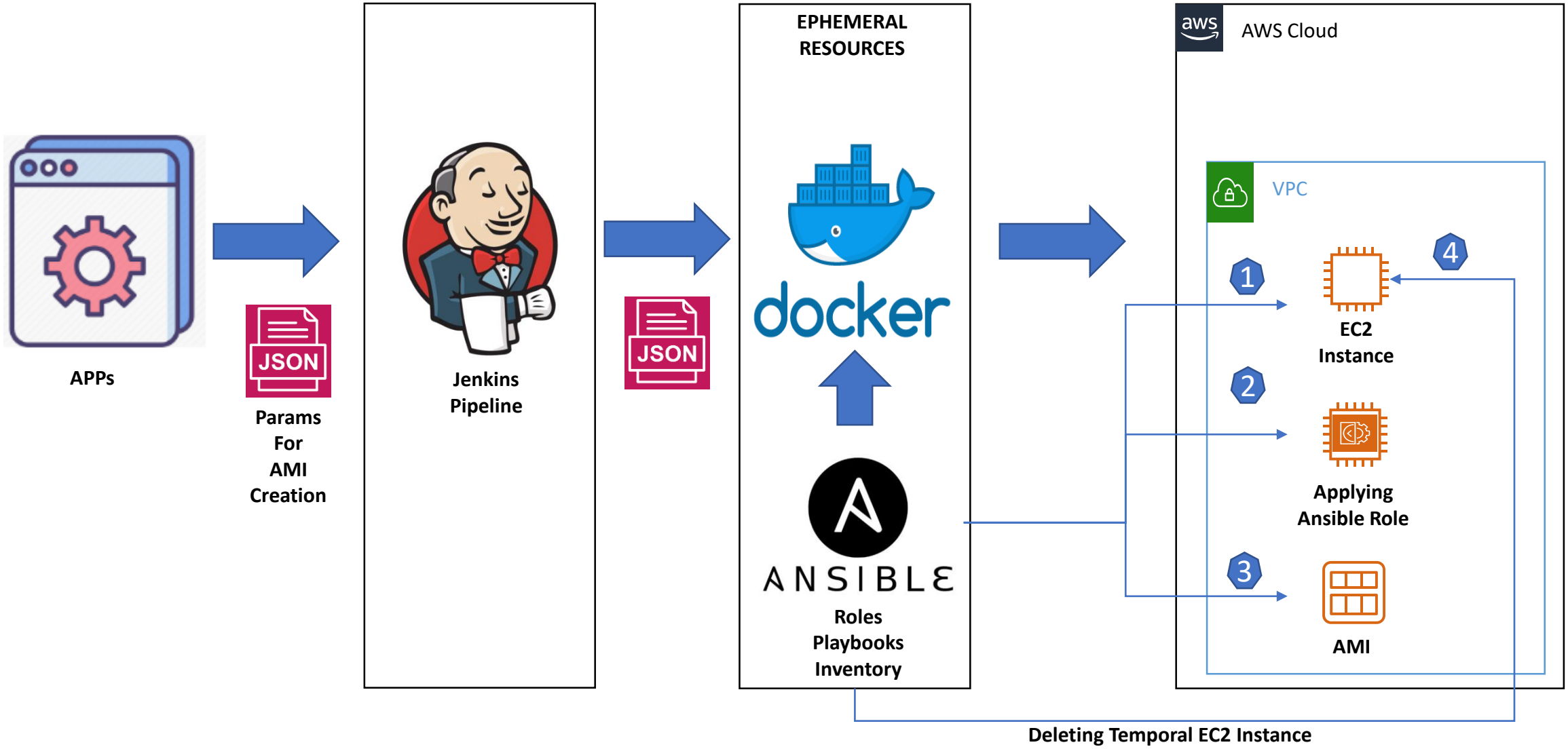
AWS AMI Creator

Alternative Method Proposal to Standarize and Create AWS AMI

Global Architecture Model



Why use these Technologies ?

- *Jenkins*:
  - Could be used as a simple *"Orchestrator"* to execute your *"Container Application" [AWS AMI Creator at this case]*
  - Centralize in the created Pipeline the AMI Creation
  - Support Params and Several Execution models with them:
    - API
    - Pipeline Invokes from JOBs/Other Pipelines
  - Support the creation of Workers from Containers (Detailed in another Proposal)

Why use these Technologies ?

- *Ansible*:
  - Declarative Language to implement “IaC”
  - *“Idempotent”*
  - *Portable / Crossplatform*
    - *Ansible Roles can be used without modification in several environments: On-Premise, Clouds, etc…*
  - *Integrated “Debug Levels and Reports”*
  - *Wide used solution in OpenSource World:*
    - *Community References and Support*
    - *Red Hat Support (If Required)*
  - Integration with AWS (More than 169 AWS Modules supported)
  - Integration with “Compliance Frameworks” [More in future Proposal]

Why use these Technologies ?

- *Docker*:
  - Isolation of the Application Dependencies
  - “IaC” – Include as additional “SourceCode” in your “Code Repo”
  - *Crossplatform – Execute your “Containerized APPs” without changes*
    - *Your local Container could be deployed in DEV/PRE/PRO without changes*
  - *“Ephemeral” and “Light”:*
    - *Destroy Containers after executions and “Save Resources”*
    - *Less resources that VMs [“Shared Storage Layers”, “Non Hypervisor”, etc..]*
  - *“Wide-Known” Solution with “Wide Opensource Community Support”*
    - *Although not used here, there are “Orchestrator” that facilitate the “Expose as Service/s”, Lifecycle, etc…*

Main Solution Components [Jenkins]

- *In the source code is included the Jenkinsfile to create the pipeline directly from it ("Pipeline as Code") – Present on root directory:*

```
.
├── CHANGELOG.md
├── Dockerfile
├── group_vars
├── inventories
├── Jenkinsfile
├── playbooks
├── README.md
├── roles
└── VERSION.md
```

Main Solution Components [Docker]

- *In our case is included in the repository a Dockerfile to create the reguired "Docker Image": - The below screenshot show the main directory for "AWS AMI Creator"*

```
.
├── CHANGELOG.md
├── Dockerfile
├── group_vars
├── inventories
├── Jenkinsfile
├── playbooks
├── README.md
├── roles
└── VERSION.md
```

Main Solution Components [Docker]

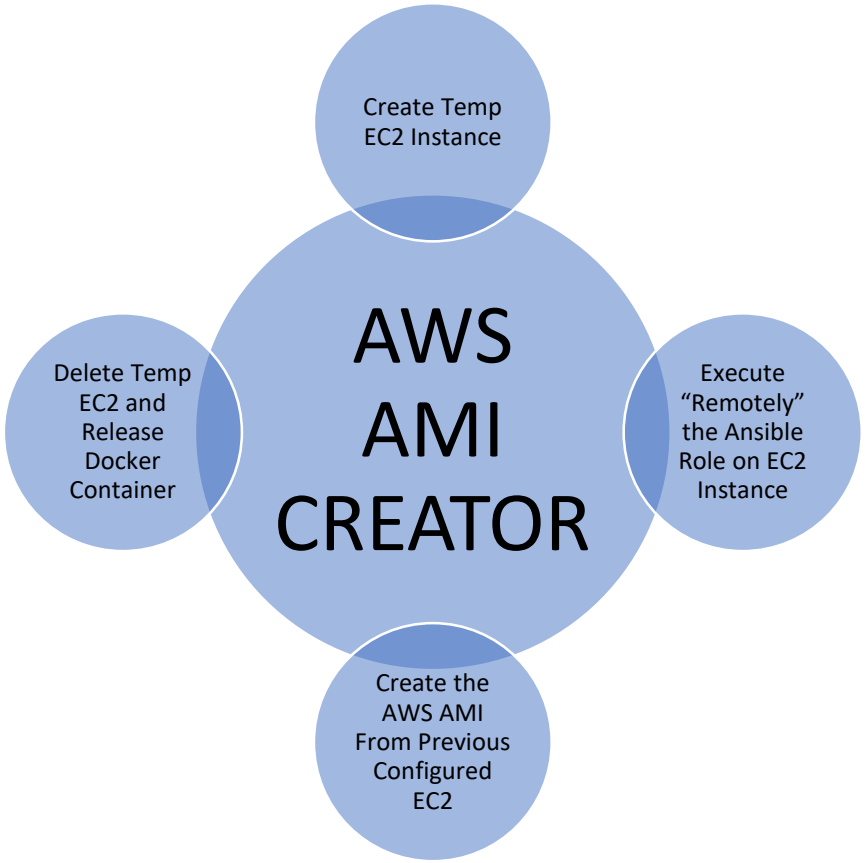
- *Using the indicated Dockerfile you can build the next Docker image:*

```
REPOSITORY          TAG                 IMAGE ID
amitool             latest              91e33ed48574
```

- *From that Image you can run the AWS AMI Creator solution given the indicated parameters*
- *Check the Repository Documentation to obtain examples about "build, run and executed" the container*

Main Solution Components [Ansible]

- *This component contains the main functionality that allows:*



Main Solution Components [Ansible]

- *How Ansible get the "work be done" ?:*
  - *Implementation of "Main WorkFlow" in the nextPlaybook:*

```
│   ├── create_ec2instanceforamibuild.yml
│   └── private_keys_ami_build
├── Jenkinsfile
├── playbooks
│   └── build_aws_ami_workflow.yaml
├── README.md
├── roles
│   ├── build_aws_ami
```

Main Solution Components [Ansible]

- *How Ansible get the "work be done" ?:*
  - *Implementation of EC2 Instance configuration in separated Roles*
    - *Currently are included under next directory:*

```
├── roles
│   ├── build_aws_ami
│   │   ├── defaults
│   │   ├── files
│   │   ├── handlers
│   │   ├── meta
│   │   ├── README.md
│   │   ├── tasks
│   │   ├── templates
│   │   ├── tests
│   │   └── vars
│   └── test_role
│       ├── defaults
│       ├── handlers
│       ├── meta
│       ├── README.md
│       ├── tasks
│       ├── templates
│       └── vars
└── VERSION.md
```

    - *Two roles included : A main workflow role and a test role (The "PLAN is include the Roles for each Desired component HERE")*

Main Solution Components [Ansible]

- *How Ansible get the “work be done” ?:*
  - *Configuration for remote connecction options and “Dynamic Inventory” in the next file:*

```
├── group_vars
├── inventories
│   ├── create_ec2instanceforamibuild.yml
│   └── private_keys_ami_build
```

  - *An from the Ansible Role corresponding to the Workflow:*

```
28
29  - name: 'Add Instance to the host group'
30    add_host:
31      #name: "{{ item.private_ip }}"
32      name: "{{item.public_ip}}"
33      groupname: 'aws_ec2_hosts'
34    with_items: "{{ ami_tmp_instance.instances }}"
35
```

Main Solution Components [Ansible]

- *How Ansible get the “work be done” ?:*
  - *The Solution Only requires the parameters related to the temporal EC2 Instance creation and the AMI Tags/Descriptions/etc… - i.e:*

```
'{"extra_sec_group" : "ami_creator_test",
   “extra_base_image”: "ami-03caa3f860895f82e",
   "extra_subnet_id" : "subnet-a45136fc",
   "extra_inst_type"  : "t2.micro",
   "extra_region": "us-west-1",
   "extra_inst_profile_name" : "aws_ami_creator",
   "extra_tags": "{}",
   "extra_key_name": "aw_ami_creator",
   "extra_ami_des" : "An ami first test with new method",
   "extra_ami_name": "ami_new_mehtod_test",
   "extra_apply_role" : "test_role",
   "extra_aws_access_key" : “<optional – AutoDetect if Included in IAM Role or another one>" ,
   "extra_aws_secret_key" : “<optional – AutoDetect if Included in IAM Role or another one>“
}'
```

VIDEO DEMO TIME

AMI Creation using test_role

Thanks for your Attention!!

- Contact Me for Any Question/Suggestion on my GitHub Site Repository:
  - https://github.com/jmbelvar81