SiNgular Application Configuration Kit - SNACK

A toolset for centralized configuration management, application building and deployment

BROOKHAVEN NATIONAL LABORATORY

a passion for discovery

U.S. DEPARTMENT OF ENERGY | Office of Science

Anton A. Derbenev, 15 June 2018

NSLS-II Controls System

EPICS

- Built on EPICS infrastructure
  - Input-Output Controller is a typical application – interfacing with hardware or other IOCs
  - IOCs are created by many engineers and require expert knowledge to build and configure
- Functions distributed over many hosts and IOCs
  - 162 Linux hosts + dozens of embedded
  - 892 IOC applications reporting on 88 hosts

The SNACK Project

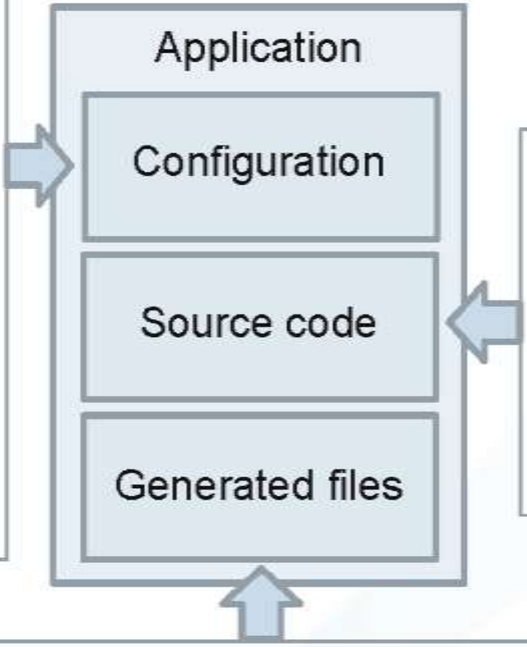
- Goal: Create a central reference for system applications, and an application deployment toolkit
  - There is no maintained index for controls applications
  - There is no uniform delivery mechanism and practice
  - Efficiency of application delivery is a strategic target by DoE requirements
- Scope: Start with Accelerator systems and IOCs
  - Vast application diversity with many unique IOCs
  - Significant solution scaling with many similar IOCs
  - Legacy applications which need accounting

Project Context

- Stakeholders: many parties and interests
  - Aiming to accommodate for needs of application developers, tool developers, system administrators, maintainers, management
- Resources: experts and environment
  - Development group, ~0.7 FTE
    - A. Derbenev, N. Maytan, summer intern joining
  - NSLS-II IT provides vital infrastructure – Ansible for automation tool, GitLab for software version control
  - Users - Controls, Ops, and Physics group members

What is an "Application"

- Parts which are unique to the application instance
  - Startup scripts
  - Files with hostname, port, IP address assignments
- Modifications to the source code
  - Git .diff files
- Configurations for external tools
  - sysv-rc-softioc "config" files

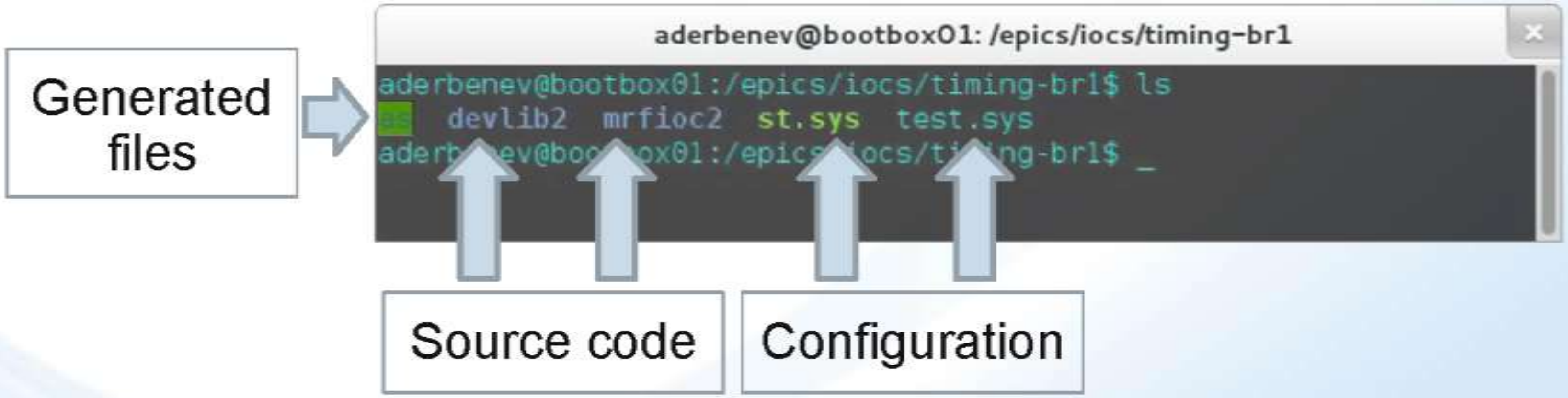


- Application sources and files which are not instance-specific
  - Driver code
  - Protocol files

- Files generated during application operation
  - Logs, backups, setting sets

Application Example – Timing IOC

- Several dozen instances running in the system
  - Source code – same EPICS driver and support library
  - Configuration – all startup scripts are different
  - Include runtime-generated files



```
aderbenev@bootbox01:/epics/iocs/timing-br1$ ls
as  devlib2  mrfioc2  st.sys  test.sys
aderbenev@bootbox01:/epics/iocs/timing-br1$ _
```

What Is a “Good” Application

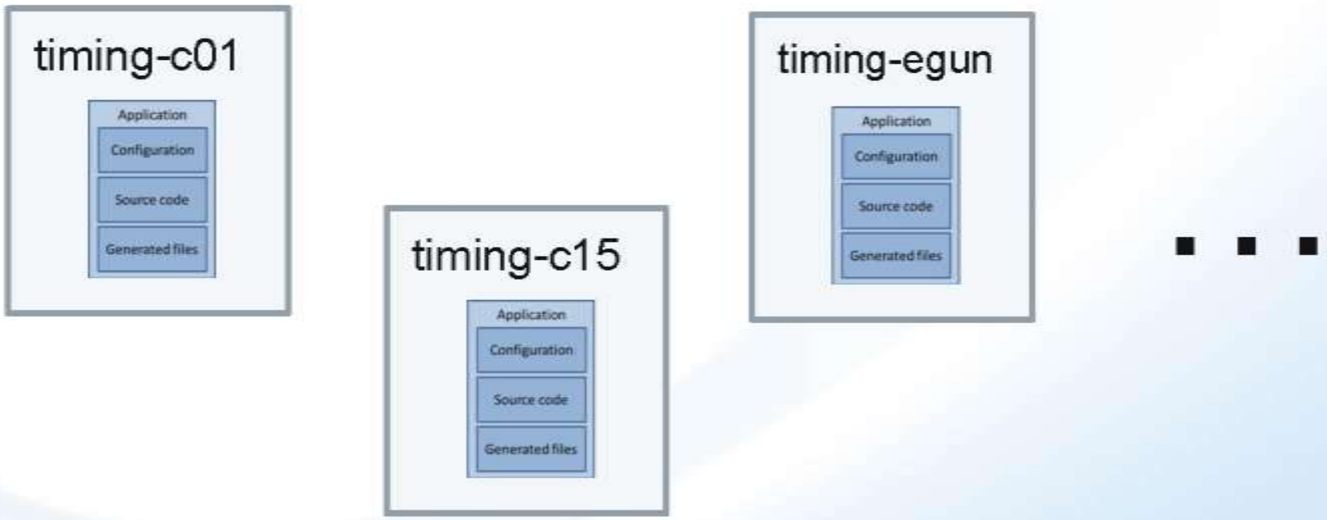
- Deployed according to practices
  - Standard installation location, files ownership and permissions, registration in the system
- Has expected structure
  - Standard configuration of the build process, known location of source files
- Is version-controlled
  - Per-application version control granularity, filtering of irrelevant files, synchronized with the repository
- Has known environment prerequisites
  - Needed system packages, external libraries, host configuration

Project Metrics

- Performance: Measuring success and system improvement
  - How many applications in the system are “good”?
  - How complete is the knowledge about the software?
- Baseline: What the current deployment approach is
  - Applications configured, compiled, run in-place by hand
  - Practices and structure are up to developers
  - Version control is not always consistent
  - Expert knowledge is crucial

Application Deployment - Now

- Applications are deployed by hand
- No central reference for application configurations
- No way to globally update source code



Analysis Of The Scope

- Possibilities: What can be improved
  - Make application inventory known
  - Make deployment process controlled and predictable
  - Reverse the reference – the system defines applications, not the opposite
- Reasons: How things are as-is
  - Different priorities during building and commissioning
  - No uniform practices and tools available
- Risks: What to keep in mind
  - New solution requires new maintenance effort
  - Can only be accomplished with experts involvement

Effort to Adopt Practice

- Application developers apply their expertise
  - Project users should learn the tool
  - Applications have to be structured
  - Deployment instructions have to be created
- Project developers provide support
  - Provide documentation and guidance
  - Manage tool host inventory and configuration access
  - Add new capabilities, improve and fix the solution

The New Solution

- Create a uniform process of software configuration, building, and deployment
  - Ansible for host management automation
  - GitLab for central configuration and tool version control

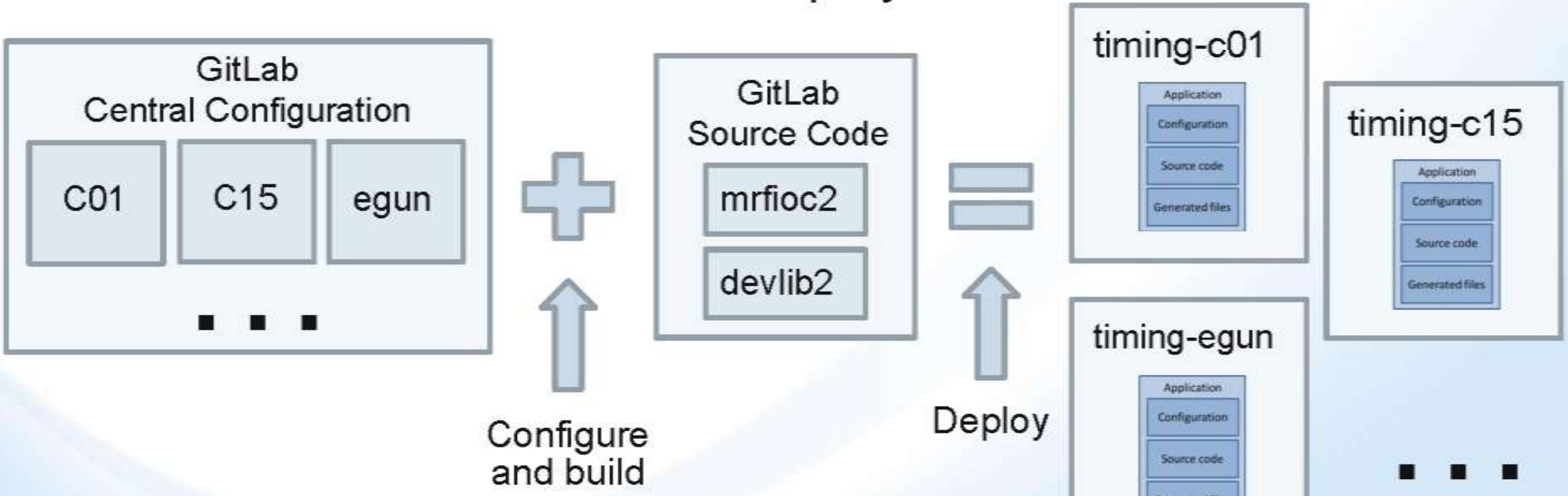
ANSIBLE GitLab

Project Timeline

- Past and present: from a simple command line tool to a powerful multi-part project
    - Initial version in August 2017, 10 months ago
    - Accessible command line interface in November
    - Top-level tool repository and function management tool added in January
    - Production tests and deployment in April-May
    - Function expansion until now
- Future: Production usage and controls applications conversion, further tool improvement
    - Hard to do during machine operations
    - Involves application experts and maintainers

Application Deployment - SNACK

- All configurations are stored in a centralized manner
- GitLab repository for every application source
- Uniform Ansible-based deployment



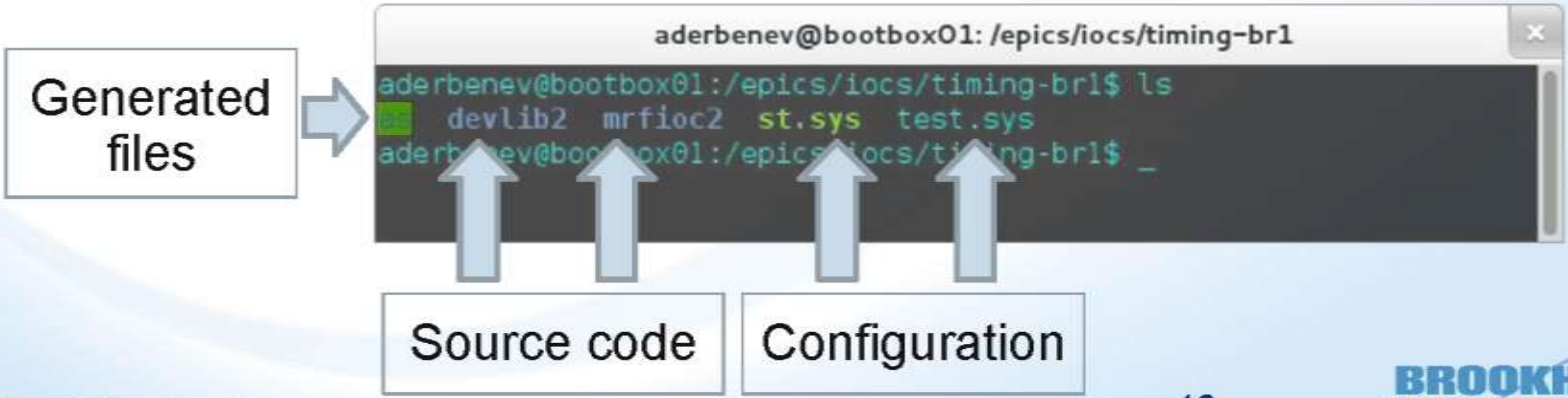
BROOKHAVEN NATIONAL LABORATORY

How It Is Prepared

- SNACK is a software delivery tool (for now)
  - Application should be created, tested, and structured first
  - Configuration should be uploaded in the central repository
  - Source code should be made available
- Developers create “recipes” to control the process
  - Specify the deployment path, file permissions, source locations, custom actions to take, generated files to keep
  - The recipe becomes a full description of how the application has to be configured, built, and deployed
  - Recipe reuse is possible via templates

Structuring The Timing IOC

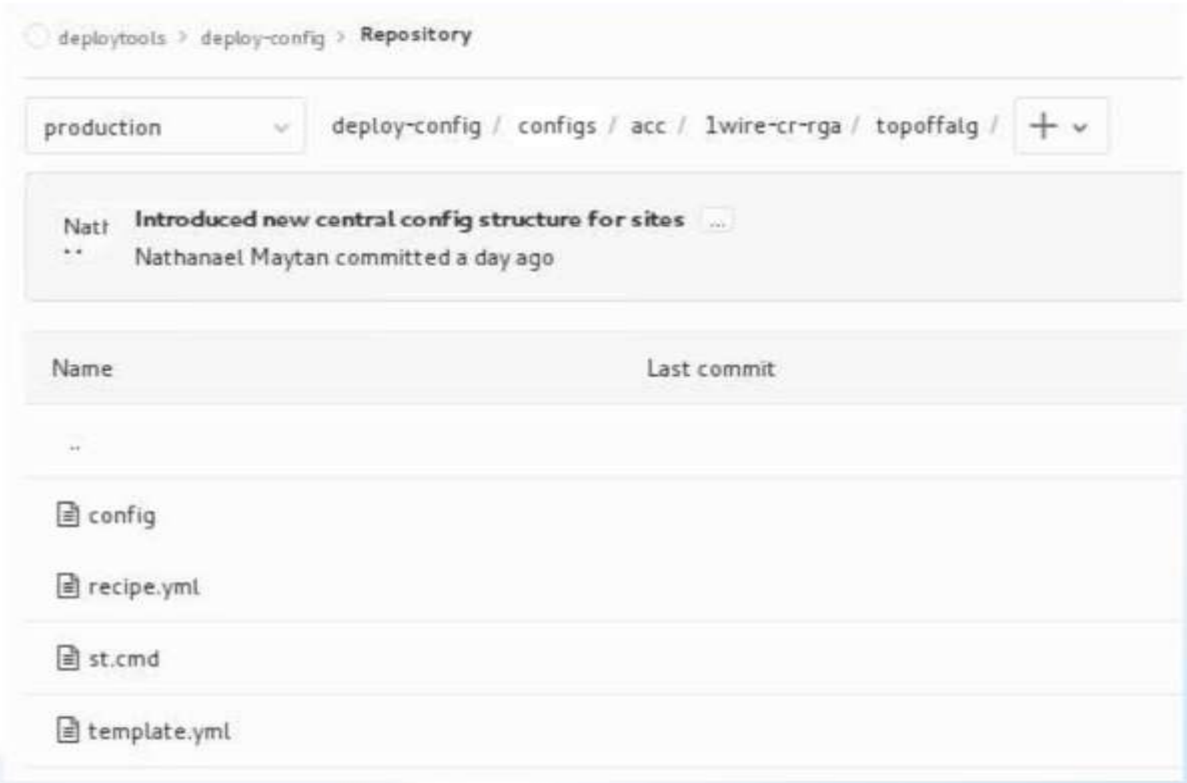
- Configuration (**st.sys, test.sys**) goes in central repository
  - https://gitlab.nsls2.bnl.gov/deploytools/deploy-config
- Source code (**mrfioc2**, **devlib2**) goes on GitLab
  - https://gitlab.nsls2.bnl.gov/accelerator/mrfioc2
  - https://gitlab.nsls2.bnl.gov/accelerator/devlib2
- Generated files (**as**) remain on application host and will be carried over between deployments



```
aderbenev@bootbox01:/epics/iocs/timing-br1$ ls
as  devlib2  mrfioc2  st.sys  test.sys
aderbenev@bootbox01:/epics/iocs/timing-br1$ _
```

SNACK Central Configuration

- Holds configurations for all apps on all hosts under all “sites”


deploytools > deploy-config > Repository

production | deploy-config / configs / acc / 1wire-cr-rga / topoffalg /

Introduced new central config structure for sites
Nathanael Maytan committed a day ago

Name	Last commit
..	
config	
recipe.yml	
st.cmd	
template.yml	

SNACK Recipe

- Convenient YAML syntax derived from Ansible
  - Flexible per-application control of the deployment mechanism
  - Reusable and supports defaulting for all options via templates

```
sources:
  - name: deploytest
    flavor: git
    get_from: https://gitlab.nsls2.bnl.gov/aderbenev/deploytest.git master
    install_in: distrib
    build_on: sandbox2
    build_method: make
    git_diff_in: distrib/deploytest.diff
```

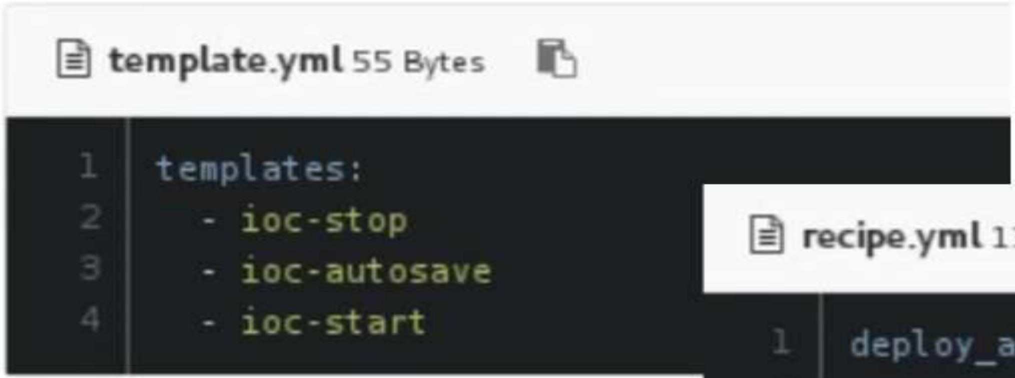
```
deploy_dir: /epics/iocs
deploy_user: softioc
deploy_group: controls
deploy_backup_enable: true
deploy_sources_enable: true
deploy_preactions_enable: true
deploy_postactions_enable: true
deploy_preserve_enable: true
```

```
post_actions:
  - user: root
    command: ./deploy-startioc.sh
    timeout: 20s
    directory: .
```

```
backup:
  directory: /epics/deploy-backups
  history: 5
```

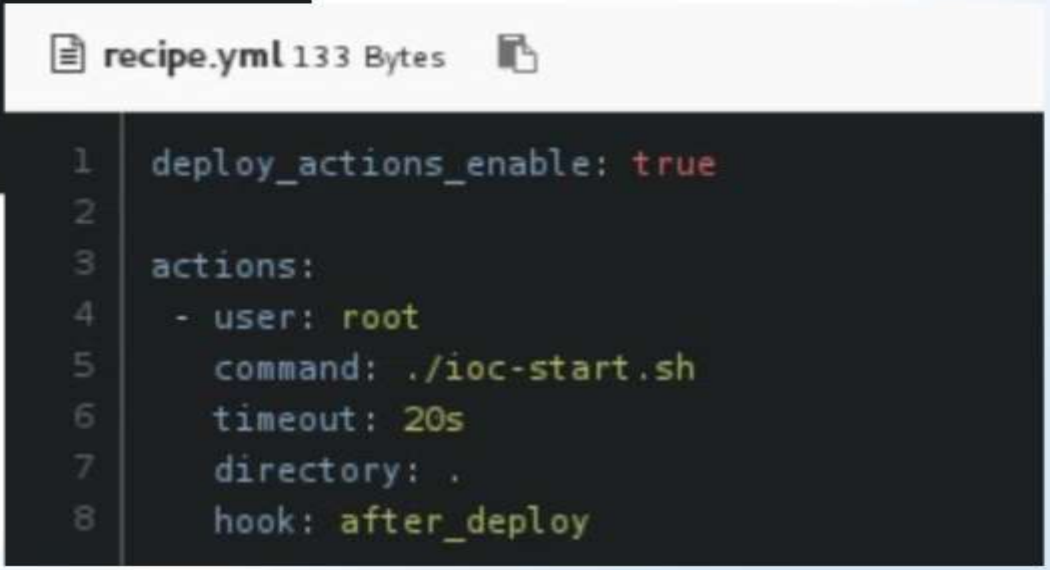
SNACK Template

- Template is a configuration not associated with a real app



template.yml 55 Bytes

```
templates:
  - ioc-stop
  - ioc-autosave
  - ioc-start
```

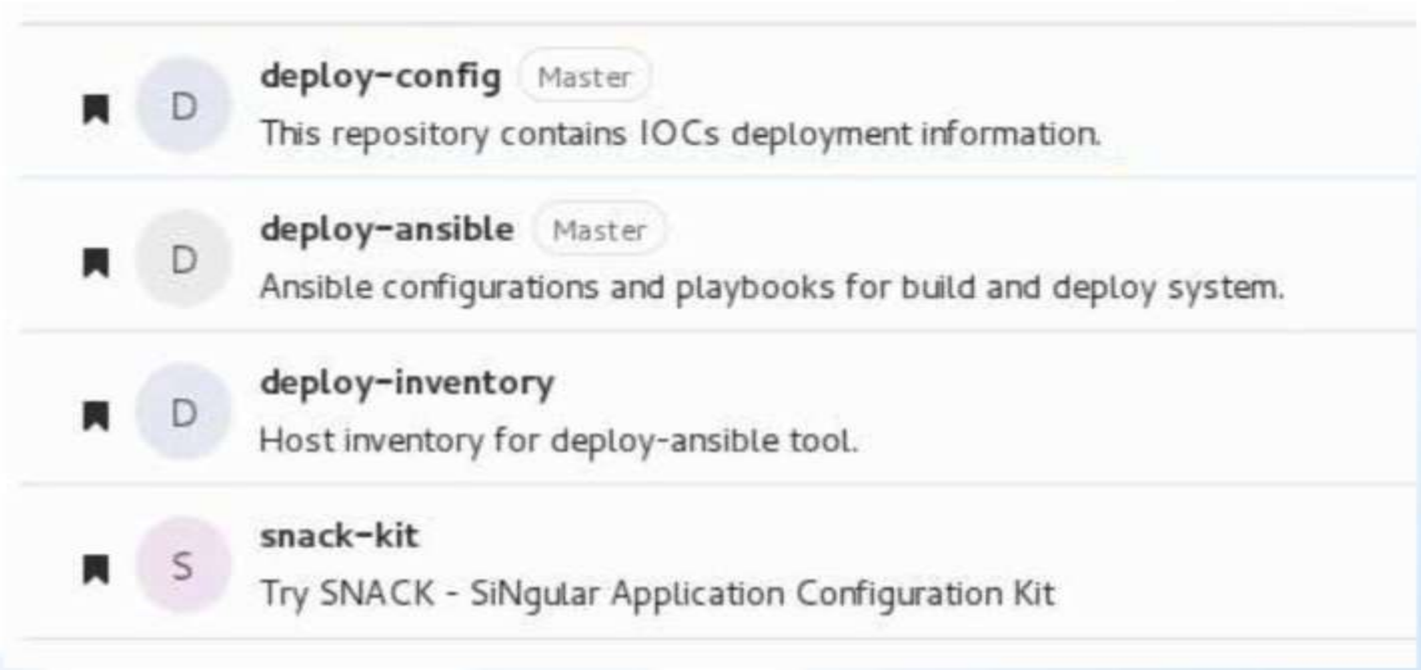


recipe.yml 133 Bytes

```
deploy_actions_enable: true

actions:
 - user: root
   command: ./ioc-start.sh
   timeout: 20s
   directory: .
   hook: after_deploy
```

SNAC Kit

- Ansible logic, Central configuration, Host inventory
- Kit utility to tie it all together



deploy-config Master
This repository contains IOCs deployment information.

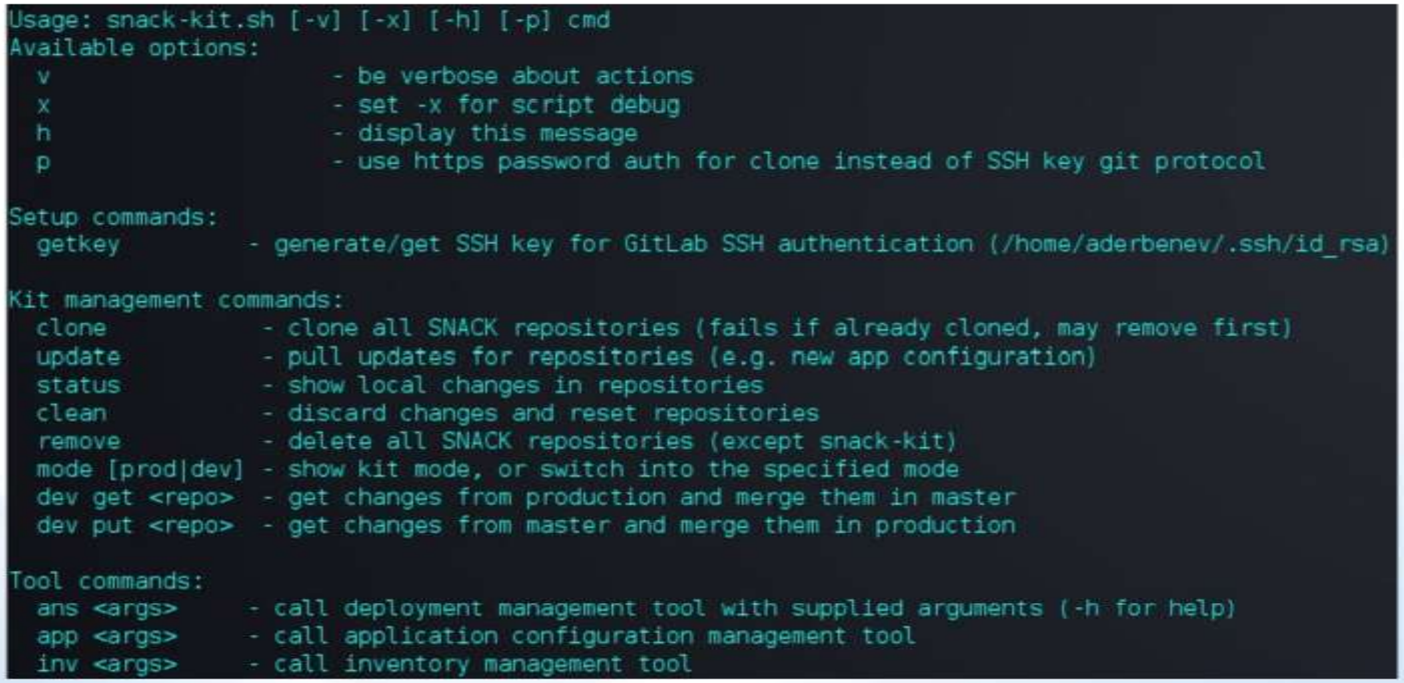
deploy-ansible Master
Ansible configurations and playbooks for build and deploy system.

deploy-inventory
Host inventory for deploy-ansible tool.

snack-kit
Try SNACK - SiNgular Application Configuration Kit

SNAC Kit Toolset

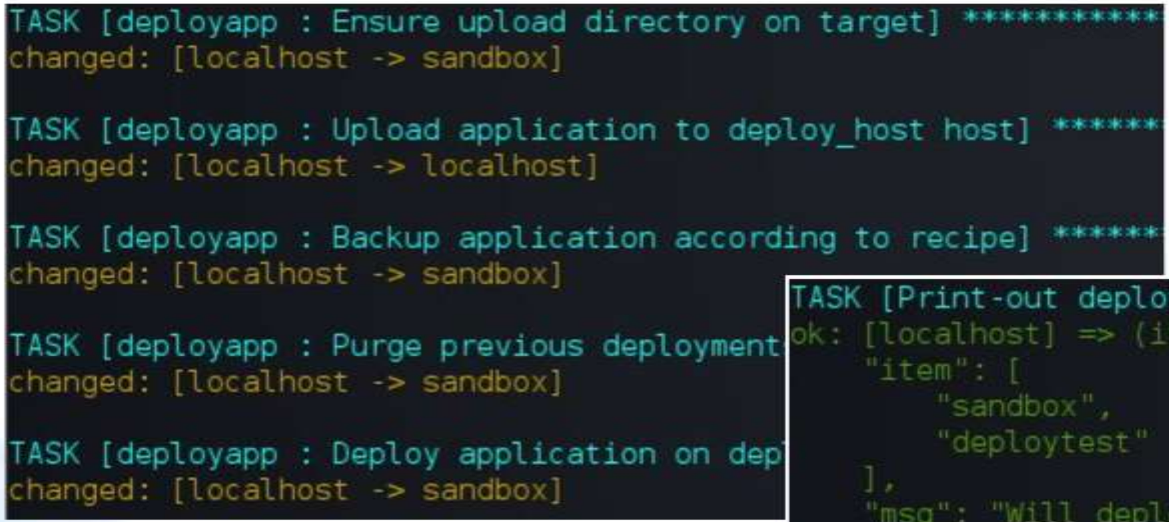
- Access to all SNACK functions
  - Project repositories management
  - Tools for deployment, configuration, inventory management



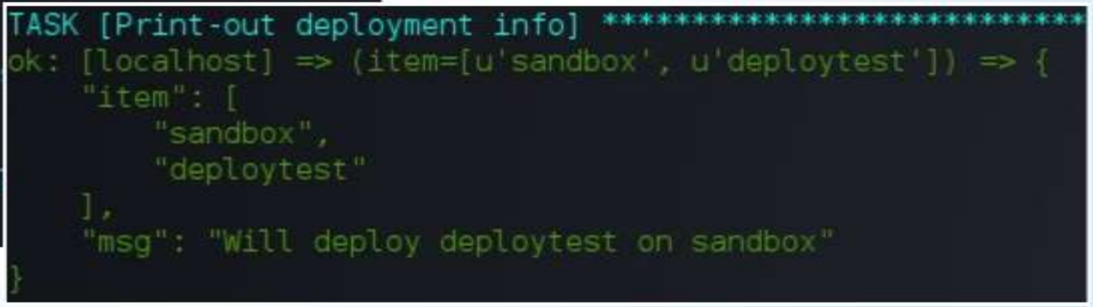
```
Usage: snack-kit.sh [-v] [-x] [-h] [-p] cmd
Available options:
  v                      - be verbose about actions
  x                      - set -x for script debug
  h                      - display this message
  p                      - use https password auth for clone instead of SSH key git protocol

Setup commands:
  getkey           - generate/get SSH key for GitLab SSH authentication (/home/aderbenev/.ssh/id_rsa)

Kit management commands:
  clone            - clone all SNACK repositories (fails if already cloned, may remove first)
  update           - pull updates for repositories (e.g. new app configuration)
  status           - show local changes in repositories
  clean            - discard changes and reset repositories
  remove           - delete all SNACK repositories (except snack-kit)
  mode [prod|dev]  - show kit mode, or switch into the specified mode
  dev get <repo>   - get changes from production and merge them in master
  dev put <repo>   - get changes from master and merge them in production

Tool commands:
  ans <args>     - call deployment management tool with supplied arguments (-h for help)
  app <args>     - call application configuration management tool
  inv <args>     - call inventory management tool
```

Deployment Process

- Sequence of many actions automated by Ansible
  - Controlled by the application recipe
  - Configures, builds, deploys, and runs the application



```
TASK [deployapp : Ensure upload directory on target] ************
changed: [localhost -> sandbox]

TASK [deployapp : Upload application to deploy_host host] ******
changed: [localhost -> localhost]

TASK [deployapp : Backup application according to recipe] ******
changed: [localhost -> sandbox]

TASK [deployapp : Purge previous deployment
changed: [localhost -> sandbox]

TASK [deployapp : Deploy application on dep
changed: [localhost -> sandbox]
```



```
TASK [Print-out deployment info] ***************************
ok: [localhost] => (item=[u'sandbox', u'deploytest']) => {
    "item": [
        "sandbox",
        "deploytest"
    ],
    "msg": "Will deploy deploytest on sandbox"
}
```

BROOKHAVEN NATIONAL LABORATORY

Project Status

- The solution is used for production deployment
  - Developers involvement is required but “old-style” applications can still run
  - Plans to use SNACK as a binary distribution solution (e.g. Base, AD)
- 200 applications in central configuration - some are simple and some are not, many legacy IOCs cleaned up
  - 51 instance of EPICS Channel Access gateways, 33+33 IOCs for rack monitoring, 20 configurations for timing, 33 for IDs, 10 for PDUs, etc.
  - Unique apps, e.g. Top Off, system status IOC, ACF switches
  - Successful cases of single and bulk deployment
- Core function is done, many improvements are implemented and more are planned
  - Further workflow simplification, more development-friendly process
  - Performance optimization, print-out clarification
  - Automation of application structuring, config management and metrics
  - Make SNACK a service?

Acknowledgements

- Controls, Ops, Physics groups for adopting
- IT for infrastructure
- Management for green light on production
- Community for inspiration

Questions?