Jenkins is for Development. Rundeck is for Operations

AND

The super short answer:
"Rundeck is made for Operations and knows about the details of your environments."

The longer answer...

Jenkins and Rundeck are complementary tools that serve two necessary -- but different -- purposes. Jenkins is a development tool, designed for automating software builds. Rundeck is an operations tool, designed for executing operations tasks.

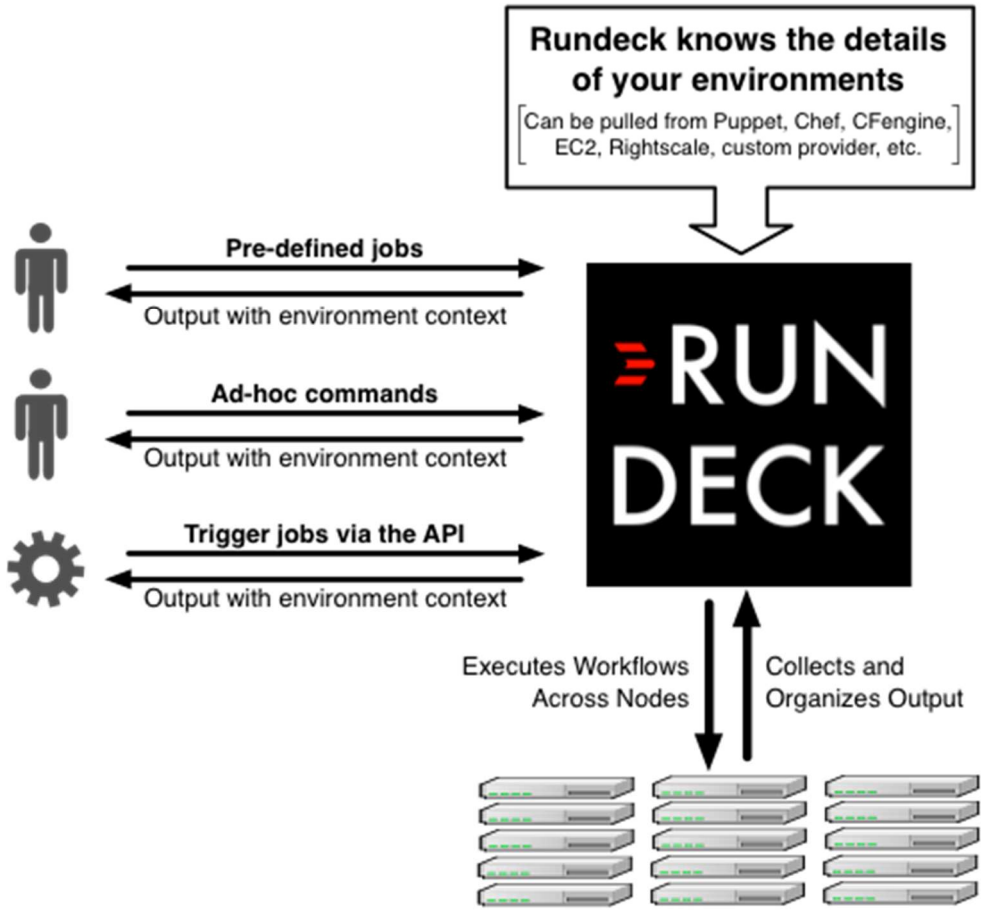
Rundeck is built specifically to turn any operations procedure into a repeatable and secure service that can be accessed securely via a Web GUI or API. Of Rundeck's features that support that operations-centric purpose, the one that stands out as the most unique from Jenkins is Rundeck's inherent knowledge of the detail of your environments (nodes, services, etc.).

Why is it powerful that Runeck knows the details of your environments?

A key concept in Rundeck's design is the idea of a Node. Nodes let you describe your infrastructure and environments giving you a view of hosts and services.

Why is this important? Most operations tasks today are going to be executed across any number of servers (from single digits to thousands). Working in a distributed environment requires a tool that knows the details of that environment, whether its for executing actions or making sense of the output coming back from all of those servers.

Where does Rundeck get its resource model information? Rundeck can pull that information from your CMDB, server inventory tool, Puppet/Chef, ec2, or any custom source to maintain an accurate resource model (even in dynamic cloud environments). You can also combine plugins to create composite views of Nodes.



Rundeck can do some really useful things right out of the box that Jenkins cannot (and you wouldn't expect from any build tool):

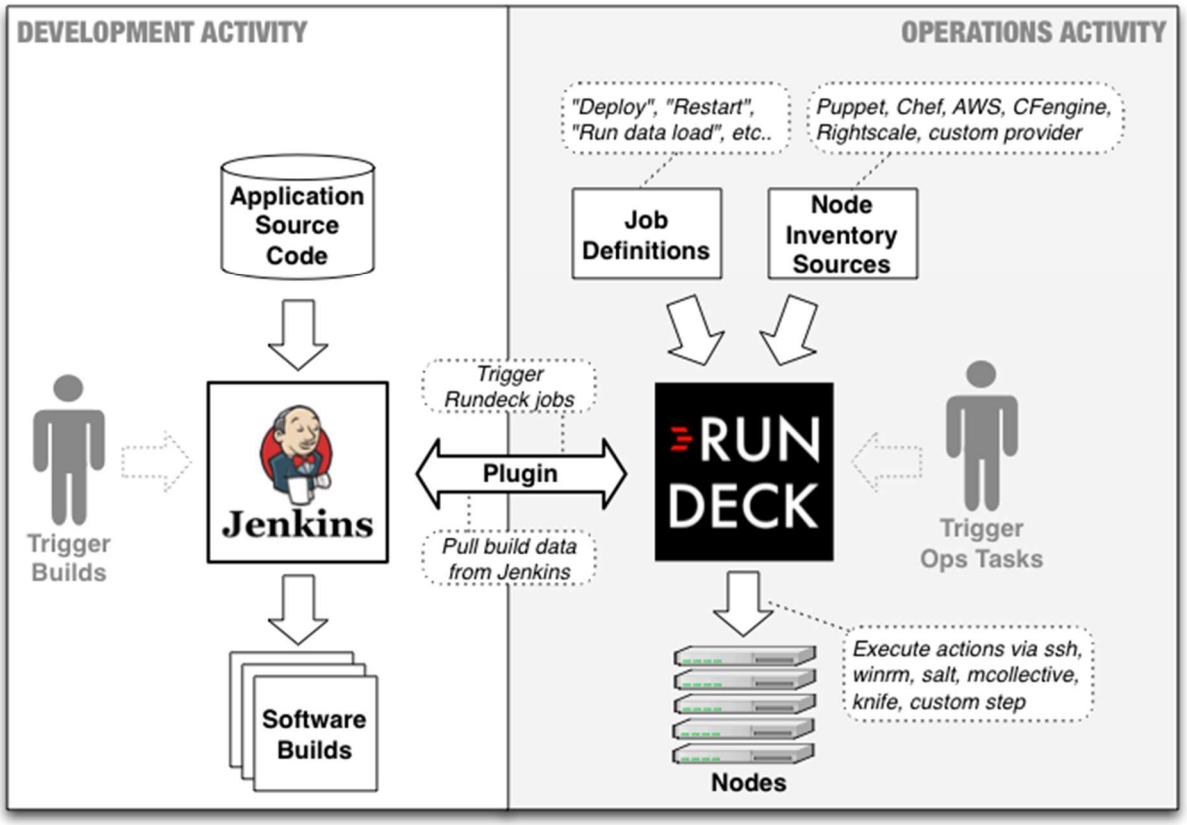
- **Rundeck executes workflows across any remote nodes/servers.** In Rundeck, you define both the workflow to be executed as well as where you want the steps of the workflow to be executed. You can set filters that determine on what nodes Rundeck will execute the job. The dispatch mechanism can be the default SSH (unix/linux), WinRM (Windows), or any other mechanism using a plugin (SaltStack, MCollective, Knife, etc.)
- **Rundeck helps you better understand the job output coming back from remote servers.** Rundeck does more than just collect the raw output from the various local or remote steps in a job. Rundeck organizes all of that output and presents it back to the user with the full context of what action happened during each job step, on what node each job step occurred, and the status of that job step. You can toggle between a workflow view or a log view collated by node and step. This is an immense benefit to both the

person designing a job and for others in your organization who need to quickly understand the job output. Many users see Rundeck's value in its role as an information radiator and as important as Rundeck's value as a workflow tool.

- **Rundeck enables you to execute ad-hoc commands in addition to saved jobs.** Often you'll find yourself in a position where you will need to execute commands and scripts in an exploratory or emergency situation. Assuming you have the appropriate permissions in Rundeck, you can use Rundeck's ad-hoc commands feature to execute actions where and when necessary (and later save them as jobs for future use or sharing with others in your organization). In this scenario, Rundeck provides a similar capability that you would otherwise need a network shell tool to achieve.
- **Rundeck makes it easy to define and execute multiple types of workflows.** By default you can choose what execution strategy you want a job's workflow to follow in order to match the real world operations challenge you are trying to solve. You can have the workflow be step-oriented (execute a step on all of the nodes selected before continuing to next step) or node-oriented (execute all steps on a nodes before moving on to the next node). You can also decide if you want steps to run in parallel and set thread count limits.
- **Rundeck jobs have built-in error handling features.** When you define your job's workflow in Rundeck you can take advantage of built-in error handling that let's you gracefully handle the various types of failures that operations teams come to expect. It even gives you the option of re-running a previous execution on only the nodes that previously failed.
- **Rundeck logs all activity and send out notifications.** The usefulness of Rundeck's built-in ability to add context to a job's output (not just what ran, but where and what the status was) doesn't end with just the person actively running or watching the job execution. Both logging and notifications are plugin points in Rundeck. This means you can send all activity information whatever logging and notification tools your organizations uses (e.g. Jabber, IRC, HipChat, Splunk, Log4J, etc). Rundeck's contextualized output is also very useful for auditing purposes.
- **Rundeck gives you access control policy that knows about your environments.** Rundeck gives you the expected capability of being able to specify who can edit or execute what jobs and when. However, because of Rundeck's knowledge of your resource model, you can also use Rundeck's fine grain ACL policies to create rules about what nodes and actions your users can do. Through LDAP integration, Rundeck can manage login access to directories like Active Directory or OpenLDAP.

How are Rundeck and Jenkins often used together?

Development and Operations are equally important parts of a technology organization. If you want to have high quality and throughput throughout your application lifecycle, you need tools that do their specific tasks well and work well together. Jenkins and Rundeck integrate well and are commonly used in the same delivery toolchain that spans from Development to Operations (and often by organizations who are looking to achieve "DevOps" style results). Jenkins will manage software builds and Rundeck will manage any and all operations tasks. Jenkins can trigger Rundeck to do deployments in a Continuous Deployment scenario. The Rundeck Jenkins plugin plays a key role in this. The diagram below shows a high-level view of a common integration scenario for Rundeck and Jenkins.



Why is it so powerful that Rundeck knows the details of your environments?

A key concept in Rundeck's design is the idea of a Node. Nodes let you describe your infrastructure and environments, giving you a view of hosts and services.

This is important because most operations tasks today need to be executed across any number of servers from single digits to thousands. Working in a dynamic, distributed environment demands a tool that knows the details of that environment to execute actions on the right subset of servers at that moment, understands the results of those actions, and takes further action based upon that understanding.

Rundeck will combine the data from multiple sources (such as Amazon EC2, Puppet, an Ansible inventory, and Device42) to allow a single view of all the information available, without requiring the maintainer of each source to adopt yet another "new standard."

Rundeck offers out-of-the-box functionality that is difficult to accomplish in Jenkins

Rundeck can execute steps across any remote nodes/servers

Because Rundeck supports many types of workflows the moment it's installed, beyond those offered by a build tool, you can execute workflows across any remote nodes/servers. There is no need to install an agent on the nodes, or for a "Rundeck team" to build and own them. In Rundeck, you define both the workflow to be executed as well as where you want the execution to occur. Filters can be used to dynamically choose servers where Rundeck executes the job based on attributes like environment, data center, operating system, application installed, and other meaningful differences. Jobs can be dispatched via the default SSH (unix/linux) or WinRM (Windows) executors, Rundeck Pro's PowerShell executor, or one of many plugins for other executors such as Ansible, SaltStack, Puppet (Bolt or MCollective), Chef (Knife), etc.

Rundeck helps you to better understand the job output coming back from remote servers. Not only will Rundeck collect the raw output from each step in a job, it organizes all of that output and presents it back to the user with full context about where, when, and why it happened. Each step's status and output are clearly visible with ANSI 256-color support, and you can format structured information like tables and JSON to be easily read and compared. Toggle easily between a workflow view or a log view collated by node and step. Even decorate the output with colorful highlights and unmistakable callouts which make it obvious even to a new employee how to find the log messages which matter the most.

These are immense benefits to both the person designing a job and others in your organization who need to quickly understand the job output. Many users achieve great value from Rundeck in its role as an information radiator, which can often be as game-changing as Rundeck's utility as a workflow tool.

Rundeck enables you to execute, track, and audit ad-hoc commands in addition to saved jobs.

In a troubleshooting situation, whether it's an emergency or you're being proactive, you will often need to execute commands and scripts in an exploratory fashion. If you have been granted appropriate permissions via Rundeck's comprehensive access control lists (ACLs), you can use Rundeck's ad-hoc commands feature to execute actions where and when

necessary. These ad-hoc commands are logged and tracked just like a Rundeck job, allowing easy collaboration, and answering questions like "what did netstat look like 10 minutes ago?" If you like, you can later save these ad-hoc commands as jobs for future use, to share with others in your organization. In this scenario, Rundeck provides a similar capability that you would otherwise need a network shell tool to achieve.

Rundeck makes it easy to define and execute multiple types of workflows.

By default, you can choose the execution strategy you want a job's workflow to follow in order to match the real-world operations challenge you are trying to solve. You can have the workflow be step-oriented (execute a step on all of the nodes selected before continuing to next step) or node-oriented (execute all steps on a nodes before moving on to the next node). You can also decide if you want steps to run in parallel and set thread count limits. With Rundeck Pro, you can also use advanced workflows to have both sequential and parallel steps in the same workflow, and to conditionally execute steps based on job settings and the output of previous steps.

Rundeck jobs have built-in error handling features.

When you define your job's workflow in Rundeck, you can take advantage of built-in error handling that lets you gracefully handle the various types of failures that a successful Operations team must be prepared for. When a job fails on some nodes, Rundeck gives you the option to re-run it only on the failed nodes.

Rundeck logs all activity and sends out notifications.

The usefulness of Rundeck's built-in ability to add context to a job's output (not just what ran, but where and what the status was) doesn't end with just the person actively running or watching the job execution. Both logging and notifications are plugin points in Rundeck. This means you can send all activity information to whatever logging and notification tools your organizations uses (e.g. Jabber, IRC, HipChat, Splunk, Log4J, etc). Rundeck's contextualized output is useful for auditing purposes.

Rundeck gives you access control policies that know about your environments.

Rundeck gives you the expected capability of being able to specify who can edit or execute what jobs and when. However, because of Rundeck's resource model's in-depth understanding of your nodes, you can also use Rundeck's fine grained ACL policies to create rules controlling access to nodes and actions based on roles. With its LDAP integration, Rundeck can manage login access via directories like Active Directory or OpenLDAP.

How are Rundeck and Jenkins often used together?

Development and Operations are equally important parts of a technology organization. If you want to have high quality throughout in your application lifecycle, you need tools that do their specific tasks well and work well together.

Jenkins and Rundeck integrate well and are commonly used by organizations who are looking to achieve DevOps-style results as a part of the same delivery toolchain that spans from Development to Operations. Jenkins will manage software builds and Rundeck will manage any and all operations tasks.

Jenkins can trigger Rundeck to do deployments in a Continuous Deployment scenario, which allows direct Jenkins access to production environments to be restricted.

In Conclusion

Jenkins and Rundeck are often used together by high performing Development and Operations teams. Using both Jenkins and Rundeck together connects repeatable build, test, and integration processes with infrastructure and production operations. With this integration, you will be able to meet security and compliance obligations while reducing the toil that impacts your ability to deliver meaningful value to the business.