	Pros	Cons	When to use	Price	Comments
Ansible	SSH-based, so it doesn't require installing any agents on remote nodes.	Less powerful than tools based in other programming languages.	If getting up and running quickly and easily is important to you and you don't want to install agents on remote nodes or managed servers, consider Ansible. It's good if your need or focus is more on the system administrator side. Ansible is focused on being streamlined and fast, so if those are key concerns for you, give it a shot.	Free open source version, with paid plans for Ansible Tower starting at $5,000 per year (which gives you up to 100 nodes).	Ansible is an open source tool used to deploy applications to remote nodes and provision servers in a repeatable way. It gives you a common framework for pushing multi-tier applications and application artifacts using a push model setup, although you can set it up as master-client if you'd like. Ansible is built on playbooks that you can apply to an extensive variety of systems for deploying your app.
	Easy learning curve thanks to the use of YAML.	Does its logic through its DSL, which means checking in on the documentation frequently until you learn it			
	Playbook structure is simple and clearly structured.	Variable registration is required for even basic functionality, which can make easier tasks more complicated			
	Has a variable registration feature that enables tasks to register variables for later tasks	Introspection is poor. Difficult to see the values of variables within the playbooks			
	Much more streamlined code base than some other tools	No consistency between formats of input, output, and config files			
		Struggles with performance speed at times.			
Chef	Rich collection of modules and configuration recipes.	Learning curve is steep if you're not already familiar with Ruby and procedural coding.	Before considering Chef, make sure you're familiar with Git, as it's required for configuration, and Ruby, as you'll have to be writing in it. Chef is good for development-focused teams and environments. It's good for enterprises looking for a more mature solution for a heterogeneous environment.	Free open source version, standard and premium plans priced on a per node per month basis that can get down to $6/node/month or $6.75/node/month respectively at high volume.	Chef is an open source tool for configuration management, focused on the developer side for its user base. Chef operates as a master-client model, with a separate workstation needed to control the master. It's based in Ruby, with pure Ruby used for most elements you write. The Chef design is transparent and based on following the instructions it's given, which means that you'll have to make sure your instructions are clear.
	Code-driven approach gives you more control and flexibility over your configurations.	It's not a simple tool, which can lead to large code bases and complicated environments.			
	Being centered around Git gives it strong version control capabilities.	Doesn't support push functionality.			
	'Knife' tool (which uses SSH for deploying agents from workstation) eases installation burdens.				
Puppet	Well-established support community through Puppet Labs.	For more advanced tasks, you will need to use the CLI, which is Ruby-based (meaning you'll have to understand Ruby).	Puppet is a good choice if stability and maturity are key factors for you. It's good for large enterprises with a heterogeneous environment and range of skills on the DevOps team.	Puppet is a good choice if stability and maturity are key factors for you. It's good for large enterprises with a heterogeneous environment and range of skills on the DevOps team.	Puppet is one of the long standing tools in the full-fledged configuration management space. It's an open source tool, but given how long it's been around, it has been well vetted and deployed in some of the biggest and most demanding environments. Puppet is based in Ruby, but uses a customized Domain Scripting Language (DSL) closer to JSON for working within it. It runs as a master-client setup and uses a model-driven approach. The Puppet code design works as a list of dependencies, which can make things easier or more confusing, depending on your setup.
	It has the most mature interface and runs on nearly every OS.	Support for pure-Ruby versions (rather than those using Puppet's customized DSL) is being scaled back.			
	Simple installation and initial setup.	Because of the DSL and a design that does not focus on simplicity, the Puppet code base can grow large, unwieldy, and hard to pick up for new people in your organization at higher scale.			
	Most complete Web UI in this space.	Model-driven approach means less control compared to code-driven approaches.			
	Strong reporting capabilities.				
SaltStack	Straightforward organization and usage once you're past the setup phase.	Difficult to set up and to pick up for new users.	Salt is a good choice if scalability and resiliency are a big concern. It's good for system administrators thanks to its usability.	Free open source version, and a SaltStack Enterprise version that is based on an annual per node subscription basis. Specific pricing is not listed on their site (just a "Contact us" link), but others have reported a $150 per node per year starting point.	SaltStack (or Salt) is a CLI-based tool that can be set up as a master-client model or a non-centralized model. Based in Python, Salt offers a push method and an SSH method of communication with clients. Salt allows for grouping of clients and configuration templates to simplify the control of the environment.
	Their DSL is feature-rich and isn't required for logic and states.	Documentation is challenging to understand at the introductory level.			
	Input, output, and configs are very consistent – all YAML.	Web UI is newer and less complete than other tool's Web UIs in the space.			
	Introspection is very good. It's easy to see what's happening within Salt.	Not great support for non-Linux OSs.			
	Strong community.				
	High scalability and resiliency in the master model with minions and hierarchical tiers.				

Fabric	Good at deploying apps written in any language. It doesn't depend on system architecture, but rather OS and package manager.	Fabric is a single point of failure set up (generally the machine you're running the deploy on)	If you're just starting out in the deployment automation space, Fabric is a good beginning point. It helps if your environment involves at least a little bit of Python.	Free	Fabric is a Python-based tool for streamlining SSH in application deployments. Its primary usage is for running tasks across multiple remote systems, but it can also be extended with plugins to provide more advanced functionality. Fabric will configure your system, do system/server administration, and automate the deployment of your app
	Simpler and easier to deploy than some other tools in this space	Uses a push model, so not as well suited for a continuous deployment model as some other tools in this space			
	Extensively integrated with SSH for script-based streamlining	While it's a great tool for deploying apps in most languages, it does require Python to run, so you must have at least a little Python in your environment for Fabric.			