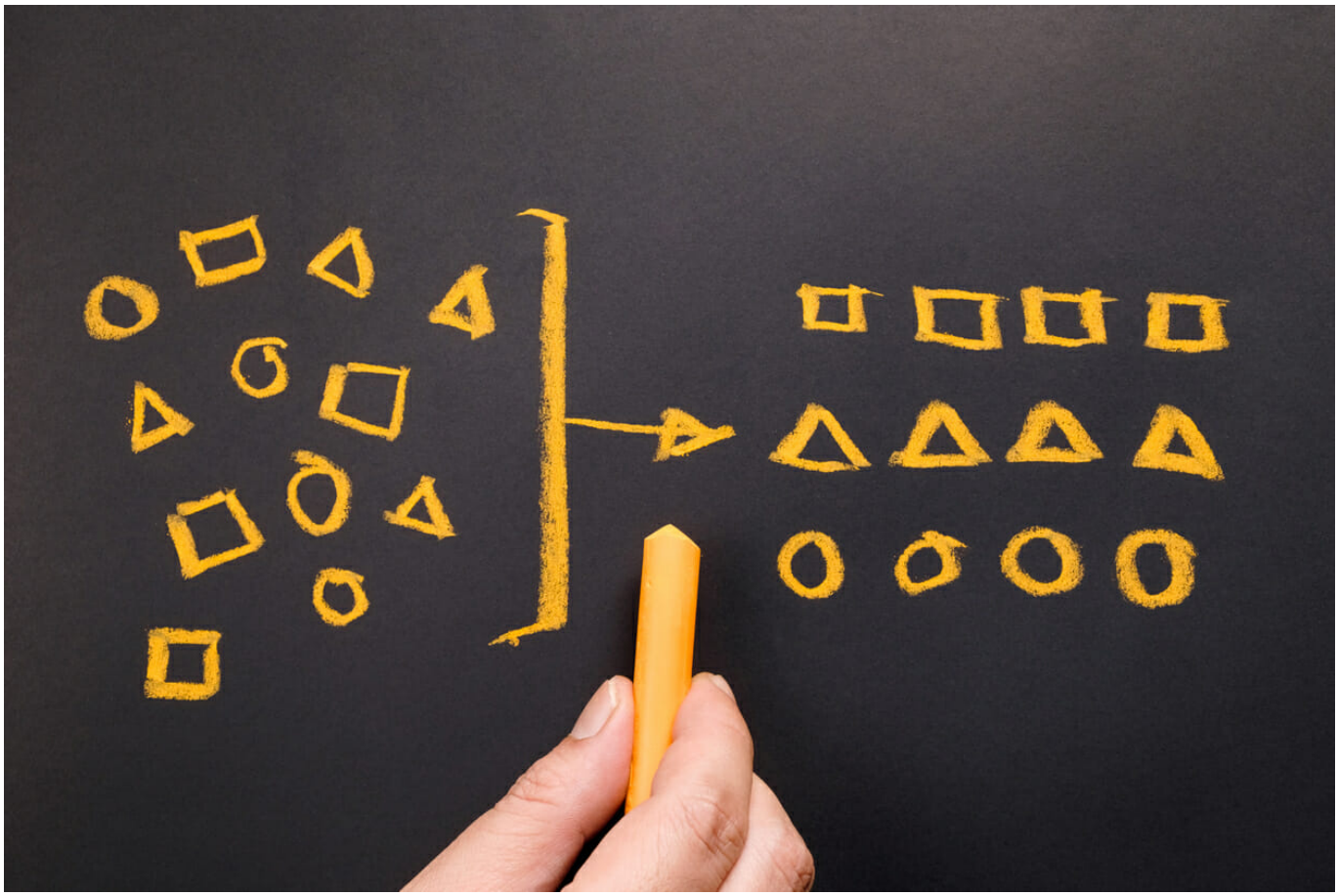
Simplify Day 2 Operations with Ansible for FlashBlade



On-premises IT operations teams manage compute, network, and storage infrastructure in various phases. For example:

- Day 0 operations include identifying the requirements, architecting the system, and designing the layout.
- Day 1 consists of installing and configuring the infrastructure.
- Day 2 enables operations, maintenance, and optimization of the infrastructure during its entire lifespan until retirement.

In the past, organizations have made efforts to automate the provisioning, consumption, monitoring, and reporting of infrastructure using bash and shell scripts in isolation. Modern data centers automate infrastructure provisioning and consumption for various business needs on-demand using automation tools like Ansible, Chef, and Puppet. These tools provision infrastructure as code (IaC) using standard templates that abstract the underlying infrastructure with RESTful APIs. These APIs are reusable for many complex and routine jobs defined in workflow pipelines.

Simplifying Automation with Ansible

Ansible is a very popular automation tool: It's easy to learn and simple to use open-source software. Ansible runs without any agent on any of the UNIX hosts. Ansible collections provide a structure that includes playbooks, roles, plug-ins, and modules. Ansible collections also give you the flexibility to fix bugs on third-party modules and connection plug-ins from vendors like Pure Storage®. You can switch the content quickly without having to wait for bug fixes in the next Ansible release.

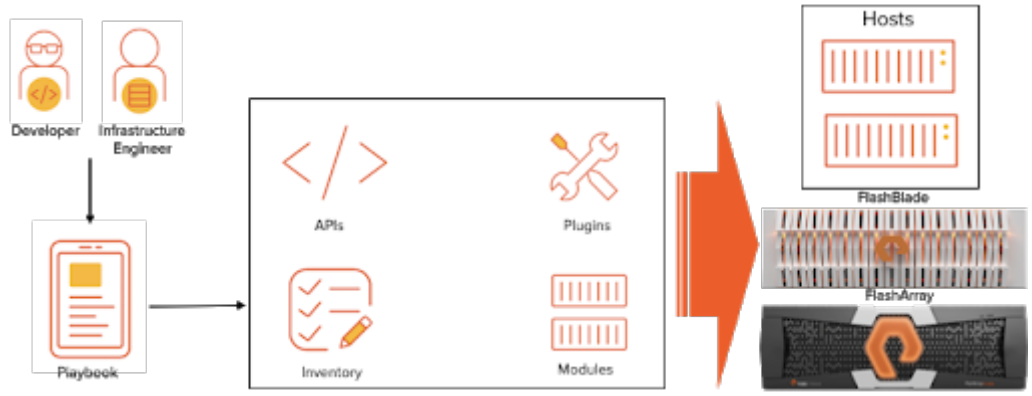


Figure 1

Ansible components consist of an inventory that manages the lists of static or dynamic hosts (Figure 1). Python-based RESTful APIs connect with storage endpoints like FlashArray™ and FlashBlade® using connection plug-ins and modules. An Ansible playbook uses the different components of the collection to perform a specific Ansible job.

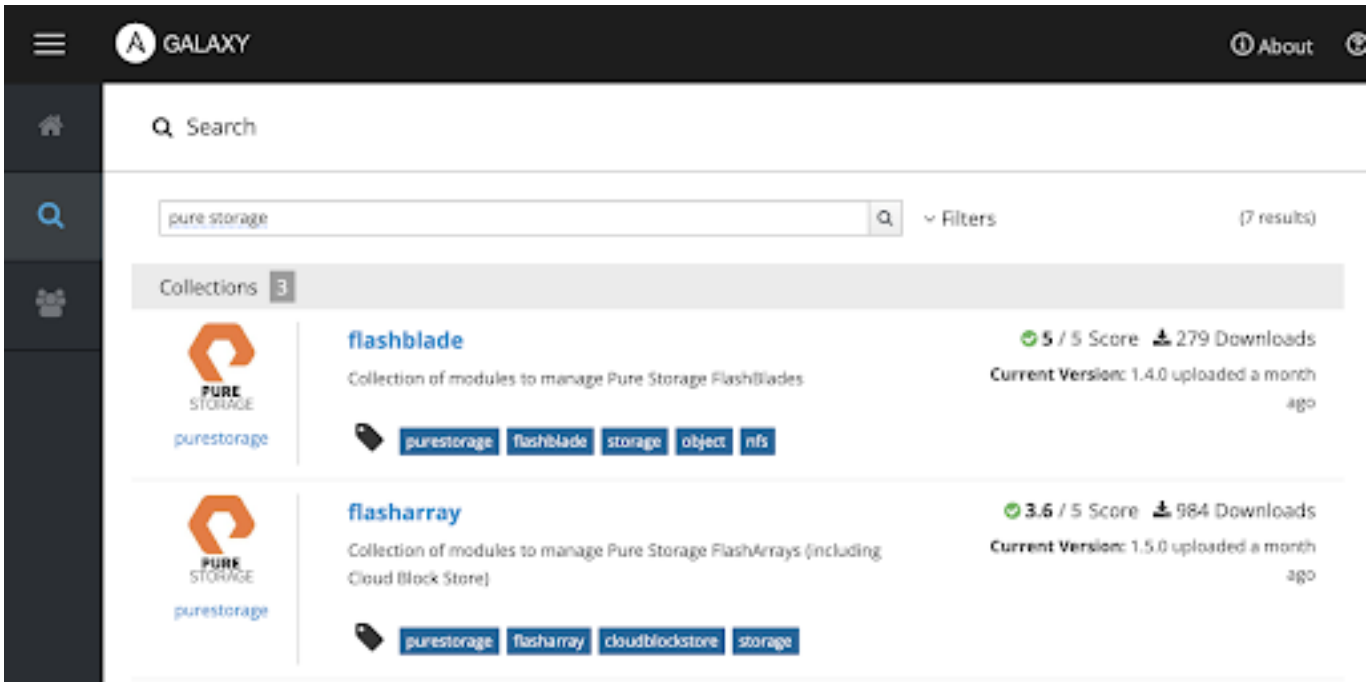


Figure 2: The collections for FlashBlade and FlashArray are now available in the Ansible Galaxy.

Leveraging Ansible Playbooks for FlashBlade

This blog post focuses on empowering developers and infrastructure admins to automate the provisioning of FlashBlade—as well as data-management capabilities like data protection and disaster recovery—for Day 2 operations using the new sample Ansible playbooks. You can find relevant Ansible playbooks on the FlashBlade GitHub page.

The new playbooks provision and configure data-management capabilities with zero storage touch for developers as well as end-user and infrastructure administrators. The playbooks perform the following functions:

- Provision
    - File systems with snapshot policy on FlashBlade
    - Multiple NFS shares along with export policies on FlashBlade (and mount them on the hosts listed in the inventory with appropriate mount options)
    - Object-store buckets on FlashBlade
- Provision and map SMB shares from FlashBlade
- Set up and configure
    - Directory services like LDAP
    - File-system replication along with failover and failback operations
    - Object replication for on-premises and to AWS S3

The FlashBlade REST client is a Python module that simplifies the integration with the FlashBlade REST interface. Pure Storage updates, publishes, and maintains libraries that are needed to get FlashBlade REST clients working. (FlashBlade API v1.11 is the latest version at the time of publishing.) The playbooks will be backward-compatible with newer versions of the APIs as they're continuously released in the future.

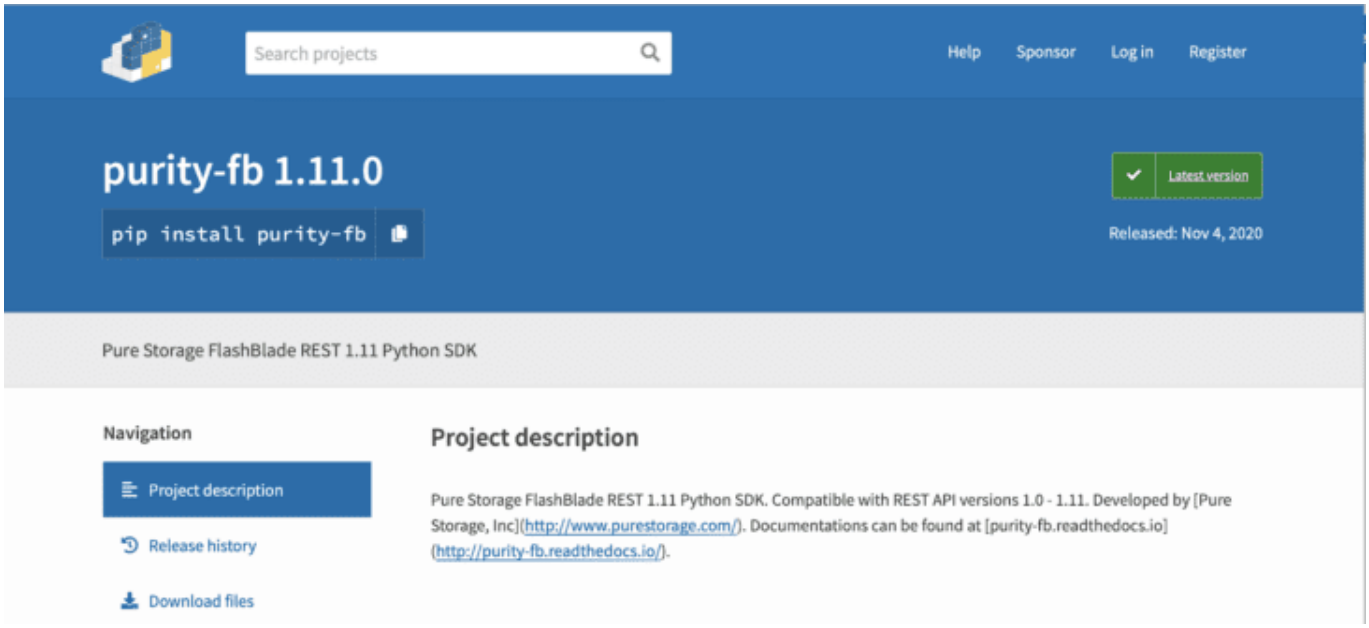


Figure 3: Pure Storage Python-based REST APIs for FlashBlade

You can find all the prerequisites for Ansible in the README section on Pure's GitHub page.

The new sample Ansible playbooks use the collection components with the roles and tasks assigned to them. The playbooks use ansible-vault to encrypt the FlashBlade API-tokens as shown in the table below. The fb_secrets.yml file is created as one of the prerequisites and is password protected. This is a one-time configuration to include all the FlashBlade devices that will be used in the environment and with all the other playbooks. You can also configure and tag the sample playbooks for FlashBlade devices in different data center locations.
[crayon-642785086dd04771904257/]
All the new sample playbooks are self-contained. For example, the file-system replication playbook takes care of setting up the connection between the replication network interfaces on the source and target FlashBlade devices. If they aren't available, the replication network interfaces on both the source and target FlashBlade devices will be created. The playbook also creates the replica link, snapshot policy, time zone, and certificates. It starts the replication between the source and target FlashBlade.
[crayon-642785086dd0d958197671/]
The above example – fb_details.repl.yml file shows that *FBServer1* is the source and *FBServer2* is the target. This playbook also can automate the failover and failback between the source and target FlashBlade devices. The file-system replication playbook has no dependencies on other sample playbooks.

By automating the provisioning and management of FlashBlade devices in data centers, you get a self-service model to consume storage IaC. Use version control management software like Git to manage various versions of the FlashBlade sample playbooks to support different teams and business processes. By leveraging the sample playbooks, developers and admins can automate provisioning and configuration of different data-management functions and streamline Day 2 operations.