BrooklynProposal

Apache Brooklyn

Abstract

Brooklyn is a framework for modeling, monitoring, and managing applications through autonomic blueprints.

Proposal

Brooklyn is about deploying and managing applications: composing a full stack for an application; deploying to cloud and non-cloud targets; using monitoring tools to collect key health/performance metrics; responding to situations such as a failing node; and adding or removing capacity to match demand.

A Brooklyn blueprint defines an application, using a declarative YAML syntax supporting JVM plugins. A basic blueprint might comprise a single process, such as a web-application server running a WAR file or a SQL database and its associated DDL scripts. More complex blueprints encompass combinations of processes across multiple machines and services, such as a load-balancing HTTP server or SDN controller fronting a cluster of J2EE application servers, in turn connected to a resilient cluster of SQL database servers. Even larger clustered application running in multiple regions can be described, with features such as message buses with resilient brokers, cacheing tiers of NoSQL key-value store servers, a high-availability database cluster and multiple application components connected across these layers.

One main benefit of these blueprints is that they are composable: best-practice blueprints for one process or pattern (e.g. a Cassandra cluster) can be incorporated in other blueprints (e.g. an application with a Cassandra cluster as one component). Another major benefit is that the blueprints can be treated as source code as part of an applications codebase: tested, tracked, versioned, and hardened as an integral part of the devops process. In some ways, Brooklyn is to run-time what Maven is to build-time.

Brooklyn knows about Chef, Salt, and similar tools, and APT and Yum and plain old shell scripts, for deploying application components. Blueprints are built from a mixture of both off-the-shelf packages such as Tomcat, MySQL, Cassandra, and many more from our library; and components that are bespoke to individual applications; together with policies that allow the application to be self-managing.

Brooklyn is built for the cloud, and will take a blueprint and deploy it to one of many supported clouds or even to multiple different clouds, or to private infrastructure (bring-your-own-node), or to other platforms. It will dynamically configure and connect all the different components of an application, e.g. so load balancers know where the web servers are and the web applications know where the database is.

Brooklyn collects key metrics to monitor the health of applications; for example, by sending a request and measuring latency, or installing monitoring tools and using those to read a server's management interface to determine the request queue length. These metrics can be fed into policies, which automatically take actions such as restarting a failed node, or scaling out the web tier if user demand exceeds capacity. This allows an application to be self-managing: to recover itself from simple failures, to scale out when demand increases and meet capacity; then scale in as demand drops and stop paying for spare capacity.

In short, Brooklyn blueprints allow the best practices for deploying and managing complex software to be codified as part of the software development process.

Background

Brooklyn is a product built from the ground up for application agility. This includes portability across non-cloud, cloud, and PaaS targets; devops-style infrastructure-as-code applied to applications; and real-time autonomic management based on promise theory. Some introductions to these concepts, associated tools, and open specifications may be useful.

Cloud computing at its core is about provisioning resources on-demand. The most widely known aspect is IaaS (infrastructure-as-a-service) such as Amazon EC2, Softlayer, Google Cloud Platform, Apache CloudStack, or OpenStack. By leveraging the Apache jclouds project (and contributing heavily to it), the Brooklyn project is able to work with a large number of such providers. Higher up the stack, however, there is an increasingly diverse set of platform targets, from PaaS (platform-as-a-service) such as Cloud Foundry and Apache Stratos, through to myriad containers and runtime fabrics such as LXC /Docker, Apache Mesos, Apache Karaf, Apache Hadoop, and Apache Usergrid and other backend-as-a-service environments. Brooklyn is based on the premise that applications may need to run in any or all of these, and the model must be flexible and open enough to support this.

The buzzword-compliant trends of agile and devops have reinforced many important lessons:

- The truth is in the code (not any ancillary documents)
- If it isn't tested then assume it isn't working
- Toolchain integration and API's are key to a project's success
- Even more critical is empowering all stakeholders to a project
- Brooklyn's focus on blueprinting and modeling as code and API's serves these principles.

Another major influence on the design of Brooklyn are the ideas of autonomic computing and promise theory. It is not necessary to have a thorough understanding of these to use Brooklyn, but contributors tend to become versed in these ideas quickly. Essentially, autonomics is based on the concept of components looking after themselves where possible (self-healing, self-optimizing, etc), and exposing a sensor (data outputs) and effector (operations) API where they may need to controlled by another element. Promise theory extends this approach by introducing the idea that communicating intent (through promises) is a more reliable basis for complex cooperating systems than obligation-based effectors. Tools such as CF Engine, Chef, Puppet, Ansible, and Salt apply promise theory to files and processes on machines; Brooklyn can leverage all of these tools, complementing it with an application-oriented model.

Finally we note some emerging standards in this area. OASIS CAMP (Cloud Application Management for Platforms) and TOSCA (Topology and Orchestration Specification for Cloud Applications) both define YAML application models similar to Brooklyn's. CAMP focuses on the REST API for interacting with such a management layer, and TOSCA focuses on declarative support for more sophisticated orchestration. Currently Brooklyn uses a YAML which complies with CAMP's syntax and exposes many of the CAMP REST API endpoints. We would like to support the hot-off-the-press TOSCA YAML and expand the CAMP REST API coverage.

Rationale

Building and deploying applications in the cloud computing era has changed many things. Provision a bare server just-in-time, and use automated tools to install an application. Use APIs to add the server to a load balancer. When load goes up, provision another server; when load drops, kill a server to save money.

Many new tools have appeared that take advantage of this new era. However each of them only solve part of the problem and don't consider the big picture. For example, configuration management tools such as Chef can, in a single command, provision a new cloud server then install and configure an application – but they require extra programming to reconfigure an load balancer whenever the pool of web servers changes. Amazon Auto Scaling can provision new servers and update load balancers, but it is dependent on CloudWatch – this means either using proxy metrics such as average response time, or writing more code to expose an application's real metrics. A dedicated monitoring tool may be able to easily monitor the key metrics with little effort, but its alerts will need to be integrated it into the server provisioning process.

So all the tools are there to to create and manage a cloud-scale application that can adapt to demand to meet user expectations without wasting money on superfluous services - but you will need several such tools and it is up to you to integrate them into your deployment plan. Some of these tools – such as the Amazon Web Services web of EC2, [CloudWatch], [AutoScaling] and [CloudFormation] – mean that you may suffer from lock-in. Related projects in [OpenStack] (Heat, Ceilometer, Murano, Solum, etc) provide similar functionality but again for a restricted target. The most common policies (such as minimising request latency) may be easy, but less common policies such as follow-the-sun and follow-the-moon [1] may be up to you to implement. Your scaling policies may understand that "high demand = add another server", but may not understand requirements such as some clustered services requiring an odd number of instances to prevent voting deadlocks.

In this context the advantage of Brooklyn becomes apparent: a single tool is able to manage provisioning and application deployment, monitor an application's health and metrics, understand the dependencies between components (such as knowing that adding a new web server means that the load balancer needs reconfiguration) and apply complex policies to manage the application. The tool provides a REST API and a GUI, and allows the autonomic blueprints to be treated as an integral part of the application. With Brooklyn, these policies become modular components which can be reused and easily added to blueprints.

[1] Follow-the-sun refers to provisioning servers in network locations close to users. User activity levels typically correlates to the time of day, with most activity during daylight hours and less during the night. Looking at a map showing daylight hours and server activity, demand patterns will tend to "follow the sun". Follow-the-moon refers to the converse, that many compute facilities will be under-utilised during the night, and cheaper pricing may be available on spot markets at these times. Therefore, the location of the cheapest compute resources will tend to "follow the moon".

Initial Goals

Brooklyn is a project in active development with some production deployments, although it is currently at version 0.7.0-M1.

Our initial goals are to ensure that Brooklyn adopts the Apache Way, to grow the community, and to continue development towards a 1.0 release which makes a much-increased community very very happy.

Current Status

Meritocracy

We firmly believe in meritocracy. Since it was open sourced in April 2012, Brooklyn has followed a published governance model based on several Apache projects where Brooklyn members have been active. Contributions can be made by anyone and are accepted after tests pass and at least one committer has reviewed it (and checked the CLAs). Contributors with a strong track record are invited to become committers and join the management committee. We welcome the opportunity for talented developers to become committers and join the PMC.

Community

Brooklyn currently has a very active but small and homogenous community. The core team is aligned to a single corporation, with just a few code contributions externally. A larger and more diverse group of individuals and companies are using Brooklyn, building blueprint projects with Brooklyn dependenies, and discussing issues on the mailing list and in pull requests. These numbers are growing slowly.

It is important that we foster and expand this community and attract more diversity to our core team.

There are two routes to growing the community which are particularly noteworthy, and both are rendered more attractive by moving Brooklyn to Apache. ISV's and other projects can engage with the community as they require blueprinting capabilities, including several Apache projects for which Brooklyn blueprints already exist (see below). Secondly, some enterprises and large organizations are already using Brooklyn and are keen to see its future safeguarded by the Apache Foundation.

Core Developers

The following core developers are proposed, being the current set of committers to Brooklyn:

Alex Heneveld, the creator of Brooklyn, is active with several open source projects (Apache jclouds, Apache Whirr, OpenStack) and standards groups (OASIS CAMP & TOSCA) in the area of application management. He co-founded Cloudsoft Corporation in 2009, where he currently serves as CTO.

Aled Sage brings over a decade's experience developing distributed applications, mostly in the enterprise sector but with an extensive list of open source project contributions. Particular areas of interest include concurrency, fault tolerance and transactional semantics.

Andrew Kennedy is skilled in programming Java applications and services using modern open-source technologies. He has experience analysing, designing and implementing applications, middleware, messaging systems and databases, and is also an accomplished penetration tester, ethical hacker, network security engineer and security analyst. He is an ASF committer with the Apache Qpid project.

Richard Downer has spent the last few years specialising in Cloud Computing. He has worked extensively with the OpenStack and CloudStack APIs, as well as Amazon EC2. Aside from Brooklyn, he has made numerous contributions to the Apache jclouds Java cloud abstraction project (http://jclouds.org).

New core developers (initial committers) for the Brooklyn project will be welcomed based on the current criteria of code contributions and broad involvement in the community through code reviews, mailing lists, and/or IRC.

Alignment

We wish Brooklyn to follow the best practices of open source, and Apache is a model of these practices. From the outset, Brooklyn has based itself on the Apache model for governance including making commits and voting on releases or substantive changes.

Brooklyn is a heavy user of many Apache projects, and Brooklyn participants have been active in discussions and code contributions with several of these communities. In particular, Apache jclouds is our most important dependency. Apache CloudStack is a reliable and often-used deployment target for Brooklyn. Apache Maven is used to build Brooklyn and in some cases to resolve dependencies at runtime. In addition, Brooklyn support has been built for deploying and managing many Apache projects (Tomcat, Cassandra, Whirr, ActiveMQ, Kafka, and many others below). These relationships are discussed further below.

Known Risks

Orphaned projects

The team behind Brooklyn firmly believe in their project and are committed to its success. They will be pushing the project until, and beyond, its community makes the project self-sustaining.

Inexperience with Open Source

Brooklyn has been open source since April 2012. All development has happened in public on [GitHub][1], backed up by mailing lists on Google Groups[2] [3] and IRC[4]. The team of committers are committed to the principles of open source, and count amongst their number existing Apache committers.

For some of the extended group of contributors more used to commercial development, the commitment to openness is a habit that still needs to be learnt. For example, there is a feeling that some things need to be discussed "internally" first. However this instinct is being un-learned as our experience with the project in open source form continues. Under the guidance of the PPMC, we do not expect this to be a risk.

[1] https://github.com/brooklyncentral/brooklyn

[2] http://groups.google.com/group/brooklyn-users

[3] http://groups.google.com/group/brooklyn-dev

[4] #brooklyncentral on Freenode

Homogenous Developers

The initial committers are all affiliated with Cloudsoft Corporation Ltd and the vast majority of the commits to the project to date have been by Cloudsoft employees. Some of the active participants are from organizations other than Cloudsoft and have expressed their support for the move to Apache and interest in continuing to contribute.

We do have a geographically distributed list of contributors. Many are in the United Kingdom, but there are a significant number of commits from people from other European countries and from the United States.

As noted previously, our community is beginning to grow and we look forward to bringing more diversity into the list of committers and PPMC members. However we recognize the current lack of diversity as a known risk.

Reliance on Salaried Developers

The initial committers are all affiliated with Cloudsoft Corporation Ltd. The vast majority of the commits to the project to date have been by Cloudsoft employees in the line of their work.

Our community is beginning to grow, and we hope to grow the community significantly and bring more diversity into the list of committers and PPMC members. However we recognise the reliance on salaried developers as a known risk.

Relationships with Other Apache Products

Brooklyn has relationships to several other Apache projects:

Brooklyn has entities for many Apache projects; for example, it can deploy instances of Cassandra, Tomcat, HTTP Server, ActiveMQ, Qpid, Solr, Storm, Karaf, Kafka, Zookeeper and CouchDB Brooklyn uses Whirr to deploy instances of Hadoop and HBase jclouds is our single most important dependency, which we use for virtually all our interaction with cloud provider APIs. CloudStack is a first-class target for Brooklyn application deployments, and we have used it on many occasions. We use Maven, HttpComponents and various Commons projects. There is some overlap with the Whirr project; at face value, both Whirr and Brooklyn allow complex applications to be deployed into cloud providers. However our emphasis is different – Brooklyn provides policy-based management to monitor applications. We consider Brooklyn to be complementary rather than competitive - Brooklyn supports the use of Whirr to deploy Hadoop clusters.

A Excessive Fascination with the Apache Brand

We consider that Apache is a natural home for Brooklyn; we license our code under the Apache License, and we make extensive use of jclouds which recently graduated from the Apache incubator; we aspire to follow open source best practices. Our proposal to the Apache Incubator is based in pragmatism, not idolatry.

Documentation

Brooklyn's primary website is available at http://brooklyn.io/

Initial Source

There is an established existing code base for Brooklyn located at GitHub: https://github.com/brooklyncentral/brooklyn

Source and Intellectual Property Submission Plan

The code is licensed under Apache License V2, and all contributions are subject to an Apache-style ICLA with Cloudsoft Corporation Ltd. In addition to the code, Cloudsoft has registered a trademark on the name "Brooklyn", and the logo currently used for the project. The PPMC will work with Cloudsoft Corporation and its stakeholders in order to identify these properties and donate them to the Apache Foundation.

There are also a number of other assets related to the project, such as its domain name, Twitter account, and IRC channel. During incubation the PPMC will identify all these assets, and arrange the transfer, replacement, or deprecation of these assets as appropriate.

External Dependencies

The vast majority of Brooklyn's direct and indirect dependencies are licensed (or dual-licensed) under The Apache License, Category A or B licenses, or are dedicated to the public domain.

A small number dependencies give cause for concern; there are two cases of LGPL dependencies, and a number of dependencies where the license is either unknown, or compatibility with Apache is unknown.

The PPMC must therefore:

- determine the license for all code where it is not currently known;
- determine ASL compatibility where this is not currently known;
- replace all dependencies that do not have an ASL compatible license.

Cryptography

Brooklyn does not directly implement cryptography code. It may rely on other projects to perform certain cryptographic tasks (e.g. calculating signatures for requests to cloud providers.)

Required Resources

Mailing lists

We seek a "dev" list to replace both the dev and user lists currently in use. In alignment with Apache's standard practices, we would also have a "private" list for the PMC members, and a "commits" list.

Source Control

We would require a Git repository named "brooklyn" to hold the Brooklyn source code, and "brooklyn-site" to hold the web site source code. We would like these to be mirrored to GitHub repositories, where we intend to follow the same model currently used by Apache jclouds.

Issue Tracking

Jira, with a project name of "BROOKLYN".

Other Resources

No other resources are required at this time.

Initial Committers

- Alex Heneveld
- Aled Sage
- Andrew Kennedy grkvlt@apache.org
- Richard Downer

Affiliations

The majority of the commits to the project so far have been made by people who were employees of Cloudsoft Corporation Ltd at the time of the contribution, and the vast majority of those commits were in the line of their employment. All named initial committers are employees of Cloudsoft.

As stated in the Known Risks section, we appreciate that this cannot continue long term, and a key goal of the podling will be to encourage people not affiliated with Cloudsoft to join the project as committers and PMC members.

Sponsors

Champion

- Chip Childers chipchilders@apache.org

Nominated Mentors

- Matt Hogstrom hogstrom@apache.org
- Alex Karasulu akarasulu@apache.org
- David Nalley ke4qqq@apache.org
- Marcel Offermans marrs@apache.org
- Jean-Baptiste Onofré jbonofre@apache.org
- Olivier Lamy olamy@apache.org
- Chip Childers chipchilders@apache.org
- Andrei Savu asavu@apache.org
- Joe Brockmeier jzb@apache.org
- Jim Jagielski jim@apache.org

Sponsoring Entity

We are asking that the Incubator sponsors the Brooklyn podling.