INTERACTIVE INTELLIGENCE

PLATFORM SERIES - VOL 1

• PLATFORM •

Hadoop the Shard!

PureCloud is a true cloud platform, but what does that really mean?

Written By **RANDY CARTER**

FORWARD

This eBook is written for anyone evaluating a prospective cloud application - many vendors say an application is ‘in the cloud’ but what really works? What should be different from traditional software? Which features should you look for, and why are they important?

*By the way, the title of this book, ‘Hadoop the Shard’, is supposed to be a joke - Hadoop is an open-source MapReduce tool from Apache, and sharding is a method of splitting a large database for efficiency.

**But we just like the way it sounds!

***And be careful if you say this to an IT person.

TABLE OF CONTENTS

What is 'built to scale?'

Cloud systems are built to scale.
But it's not done by copying an entire monolithic software stack and adding virtualized servers. You have to start building systems from objects that can be massively paralleled. These paralleled objects, derived from a Services Oriented Architecture (SOA) design are called microservices; each containing a bundle of communications, data operations, and technology that can work independently on requests. There are many types of scaling...

Scaling microservices instead of monolithic applications gives you the ability to dynamically adjust any area that needs more resources, so you can spin up a lot of resources to refactor a database or run a complex custom report without affecting normal customer response speeds.

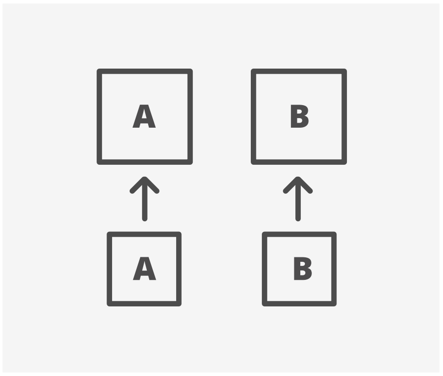


Fig 2.1 Vertical Scaling
Make boxes bigger

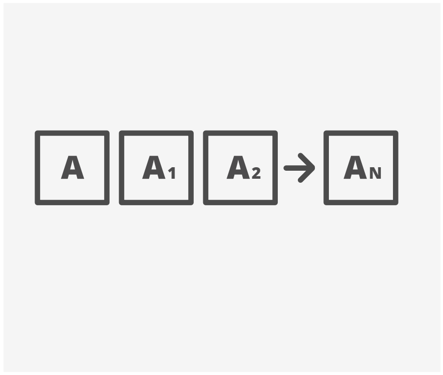


Fig 2.2 Horizontal Scaling
Make many more boxes

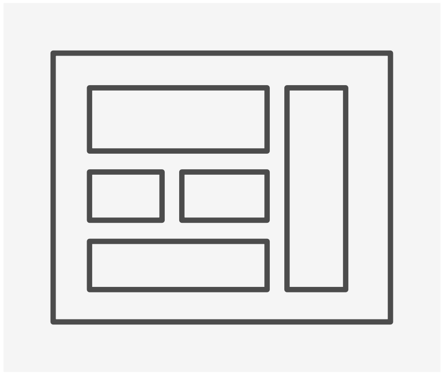


Fig 2.3 Monolithic/Scaling
Scale by adding resources, solve issues with interdependencies as they are found - or by duplicating and reconciling synced data

What are Microservices?

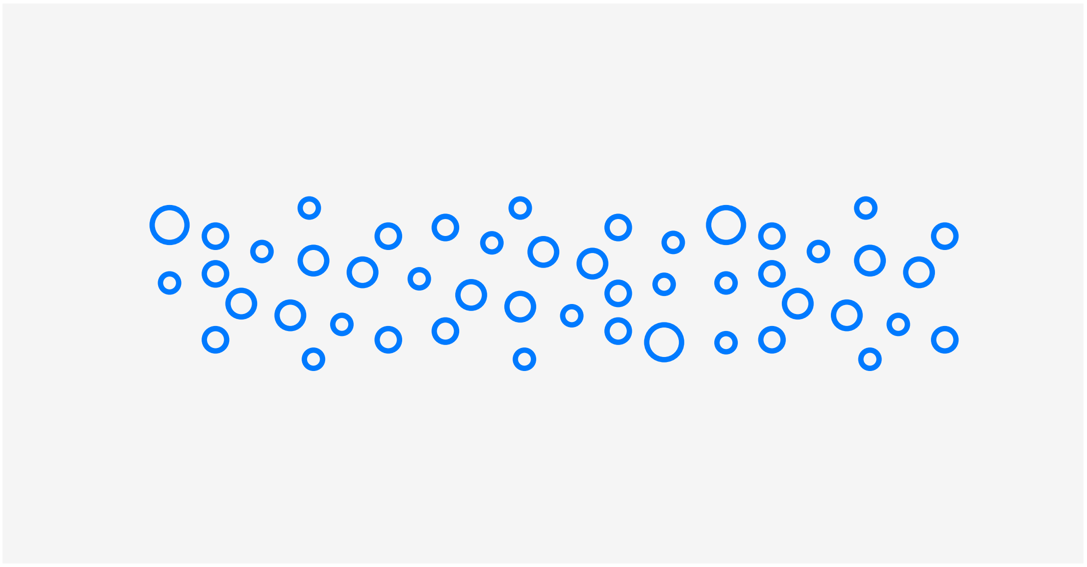


Fig 3.1

An approach to Microservices

- Modularity comes from componentized services
- Scaling tends to be horizontal
- Stateless whenever possible
- Easier to ramp-up new developers
  (or new features!)
- But... services can be harder to edit -
  When to add complexity for a new feature?
  When to split into separate microservices?
  (Strategy: build both and collect data)

How/When/Why does it scale?

Amazon and other cloud vendors sell you virtualized servers with communications bandwidth and some monitoring services. The better vendors also have integrated load balancing with a distributed architecture where all data is replicated across multiple data centers. This gives the system several options for grabbing whatever data it needs to bring everything back to the state it was in before a failure. Cloud services should not be dependent on a single hard drive and whatever its processor and connectivity limitations are.

Amazon and other cloud vendors sell you virtualized servers with communications bandwidth and some monitoring services. The better vendors also have integrated load balancing with a distributed architecture where all data is replicated across multiple data centers. This gives the system several options for grabbing whatever data it needs to bring everything back to the state it was in before a failure. Cloud services should not be dependent on a single hard drive and whatever its processor and connectivity limitations are.

Elastic Load balancing distributes requests to the independent microservices and will detect and re-startany requests that hang. This means that when there is a failure, the system will scale and to recover so quickly that users will not even notice! The same elastic load balancing is used to roll out new features, upgrades, and maintenance releases through the pools of microservices, which means that service will not be interrupted during updates. A well-architected cloud system should NEVER need to go down for maintenance!

What is Scaling?

Increased Resources = Increased Performace

Fault Tolerance

Operational Costs diminish as scale increases

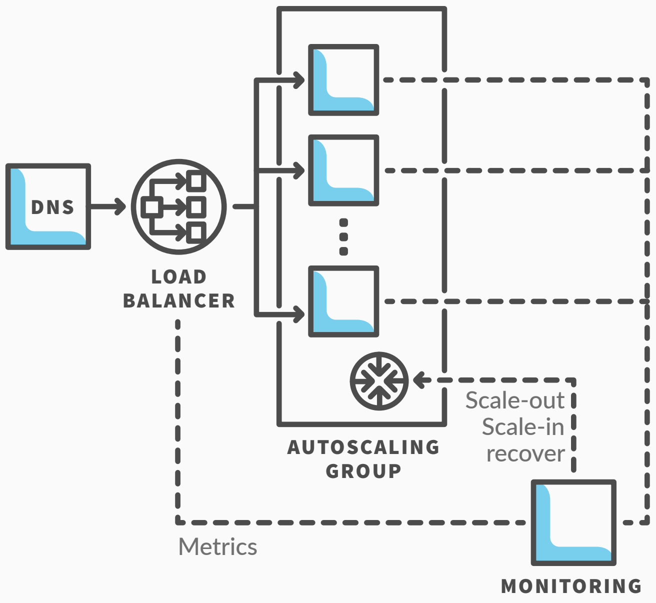

DNS
LOAD BALANCER
AUTOSCALING GROUP
Scale-out
Scale-in
recover
Metrics
MONITORING

Fig 4.1

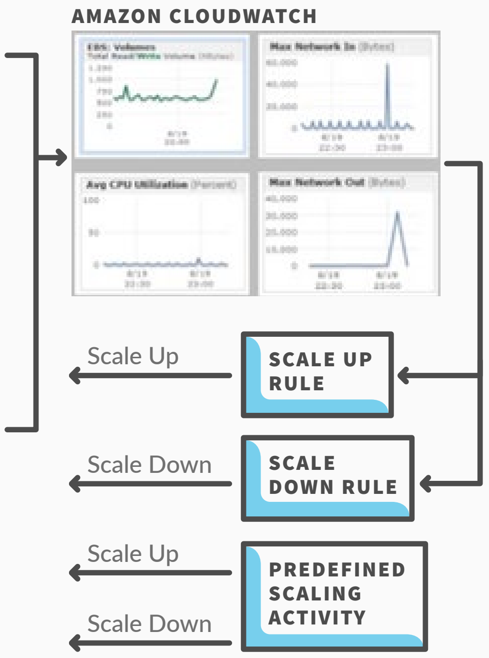

AMAZON CLOUDWATCH
EBS: Volumes
Max Network In (Bytes)
Avg CPU Utilization (Percent)
Max Network Out (Bytes)
Scale Up
SCALE UP RULE
Scale Down
SCALE DOWN RULE
Scale Up
PREDEFINED SCALING ACTIVITY
Scale Down

Fig 4.2

Public APIs, services contracts, and security...

In front of the pools of microservices are Public APIs that are the interfaces for all our clients - browsers, desktop, tablet and phone apps. Major systems have different APIs to give freedom to client developers to optimize data updating based on user tasks.

The same Public API services are available to any authenticated business process you want to build, with full developer documentation online. APIs should be versioned and updated as new features are added.

Public APIs function as a services contract between cloud services and any web access; so you will always have reliable data if you are an authenticated user.

For the vendor, the API services contract also gives cloud developers the freedom to update internal technologies at any time as long as the service continue to perform.

All data should be protected by layers of encryption - data encryption (of course) plus all requests must be HTTPS with an internal encrypted requesting user and org ID that cannot be spoofed.

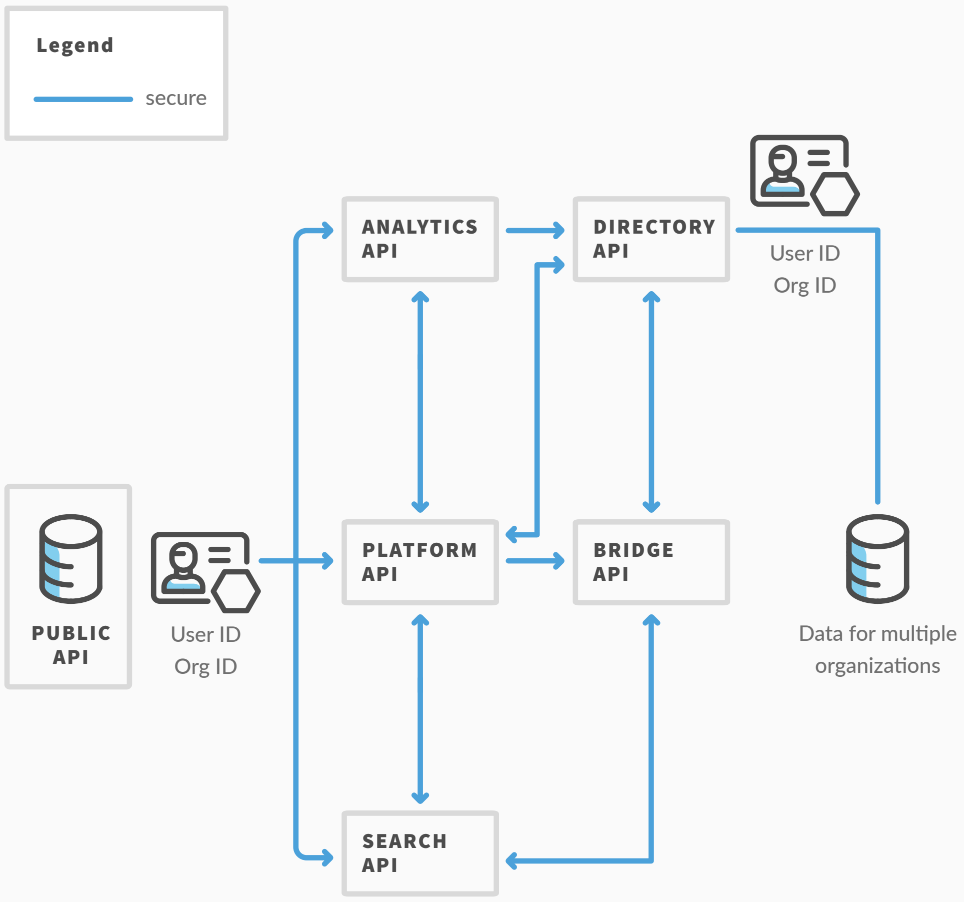


Fig 6.1 REST APIs and SECURITY
Cloud vendors also support 'Regions' that can restrict your data to a specific country for privacy compliance.

Big Data - Hadoop and the MapReduce Revolution

This one is pretty easy - there has already been a lot of coverage for the advantages of 'Big Data'. Every twenty years there is a fundamental change in database technologies, from flat files > to relational databases like SQL > and now to 'noSQL' running on document and other storage databases.

noSQL and Hadoop (Hadoop is an open-source MapReduce) are different because they are built to scale. I nstead of loading giant tables into memory and executing complex operations, MapReduce uses a pool of simple and small resources to load scraps of data and build 'graphs' of simple relationships - 'is it bigger than a breadbox?' Mapped comparisons build graphs of items that are bigger and smaller than a breadbox, and then Reduce operations clean up those graphs and find associations between graphs.

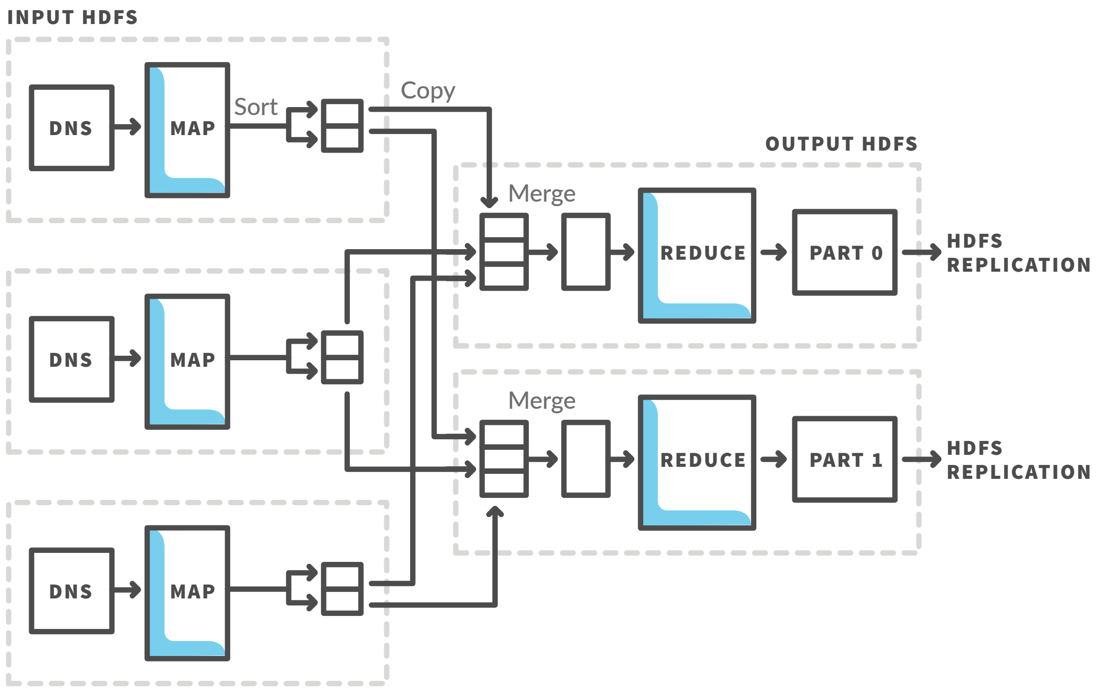


Fig 8.1 HADOOP MapReduce
Pools of MapReduce comparisons run all the time, or just whenever there is a change

Give me Continuous Deployment, or give me Pain

Continuous Delivery is Agile, keeping teams development productive. By breaking down complex work into small iterations and enforcing that each step must be completely shippable, Agile teams bite off big changes in small steps, deploying each step as they go. Continuous testing and feedback keeps teams moving forward. Smaller feature changes mean smaller tests, and smaller tests are easier to add to test automation. Automated testing keeps your development team from ever hearing "stop development so we can manually test this."

Cloud-based systems absolutely must have automated tear-down and deployment for server updates. You don't want developers picking and choosing dependencies for updates, you just want them to build contained deployable features that can be rolled out with little to no disruption.

For an Agile cloud organization, every build is a deployment; we have hundreds of deployments going on all day long just for developers to see their new code at work. We constantly deploy new features into production by rolling updates across pools of servers with no interruptions for customers.

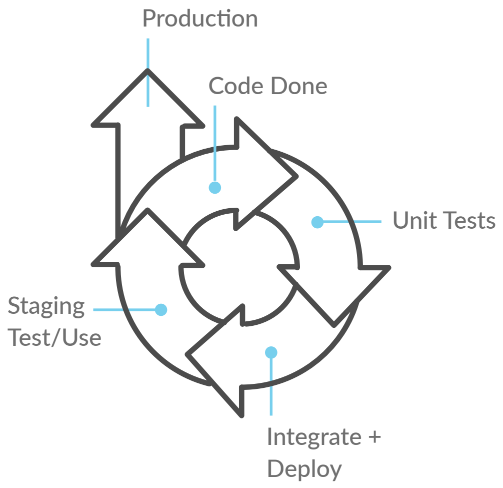


Fig 9.1 AUTOMATION
Continuous Delivery requires a very polished updating and testing approach.

Agile 'symptoms' in organizations that build cloud systems also include:

Resistance to manual processes in favor of generated process and documentation from their everyday tools.

Use of open source code for common technologies to focus on useful features Heterogenous code bases that support 'best tool available' approaches.

Willingness to try out competitive solutions in staged or production environments and then use data to drive technology decisions.

Designed for Failure?

Things fail. Software. Hardware, Networks. People.

Your customers don't care that you are having a problem, they just need your organization to help them. You should not have to care about your cloud underlying technologies either.

It was wrong to believe it was possible to build big complicated software systems that were reliable in the first place. After spending forty years trying to build around mean-time-between-failures with mountains of testing it is time for another approach. Instead, we need to design to recover from failure, and build that recovery capability into every layer of a system so it can recover gracefully from something breaking unexpectedly.

With elastic load balancing and the right level of monitoring you can detect when jobs 'fall over' and hand off their tasks to another microservice.

With a cloud infrastructure you can dynamically add and remove resources as needed. The recovery process requires a spike in resources, but with bandwidth access on demand, you can handle recoveries smoothly without slowing everything else.

Finally, with a distributed architecture where all data is replicated across multiple data centers your systems have multiple options for recovering state information even when major system failures happen.

"Everything fails,
all the time"
- Werner Vogels, CTO amazon.com

Stateless Microservices	+
Performance Monitoring	+
Elastic Load Balancing	+
Distributed Data	=

Systems that can recover
before a human user will notice

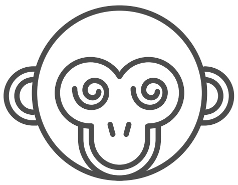


Fig 10.1 CHAOS MONKEY
But to test recovery you need to randomly knock processes over all the time (so developers find and fix state-recovery problems continuously) such as the Simian Army 'chaos' tools

Invade the other Guys' turf

Cloud systems should connect to data anywhere, and be able to do business with themselves (by providing Public API access to any useful data it crates or collects).

Organizations have existing systems that support them very well, so a good cloud system makes it easy to connect to that data wherever it may be - in other 3rd-party Cloud Applications, or in on-premises data center applications.

Connecting to data is a moving target, so tools should be lightweight and easy to use and understand.

You should be in control of syncing data. You should be able to decide which directions data is synced and which systems can update other systems.

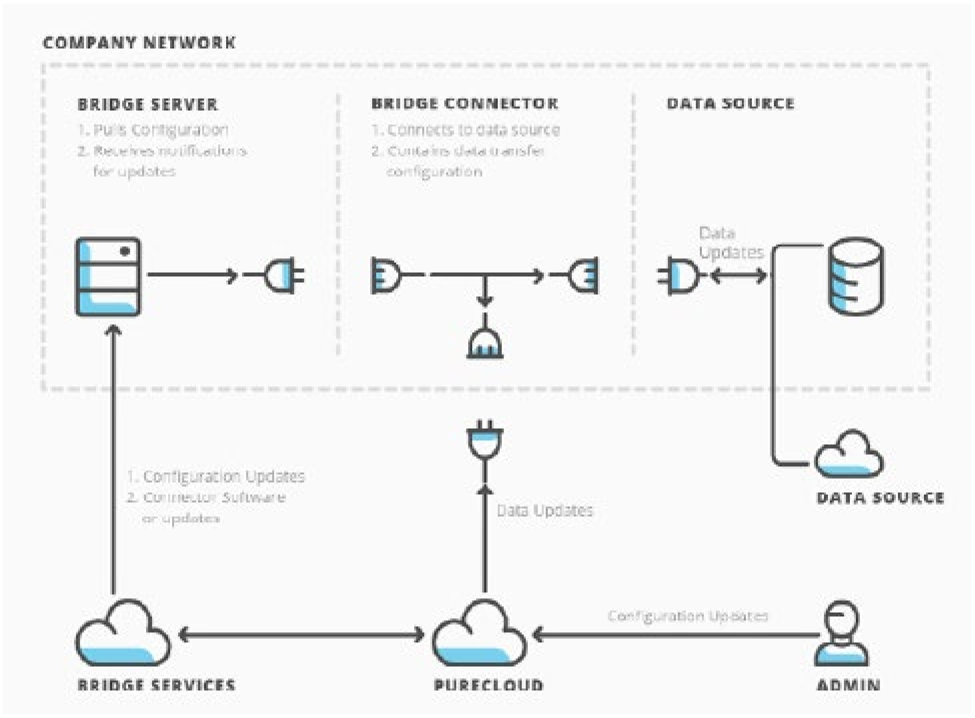


Fig 11.1 BRIDGE INTEGRATIONS

Cloud Services can sync Data Cloud-to-Cloud with other Public APIs, or Cloud-to-On-Premises via a secure local server with local authentication.

Store Everything?

With Cloud services, storage is cheap and getting cheaper. With on-premises servers, storage and backup was always a fixed capital expenditure that could only be depreciated and controlled. The cost of on-premises storage meant that software had to be designed to reduce storage - they had to decide which data was most important to save and throw away the remainder.

Cloud services let you take advantage of the relentless drop in storage costs over time. As one ofour software architects tells it “We’re a logging system that happens to do a few transactions.” We save everything, including most raw event requests, in long term storage. We use listeners on the event request stream to hook out data needed for reports to save to separate reports databases. This allows us to get smarter over time, and then go back and ‘refactor the history’ from the original events when we add new reports.

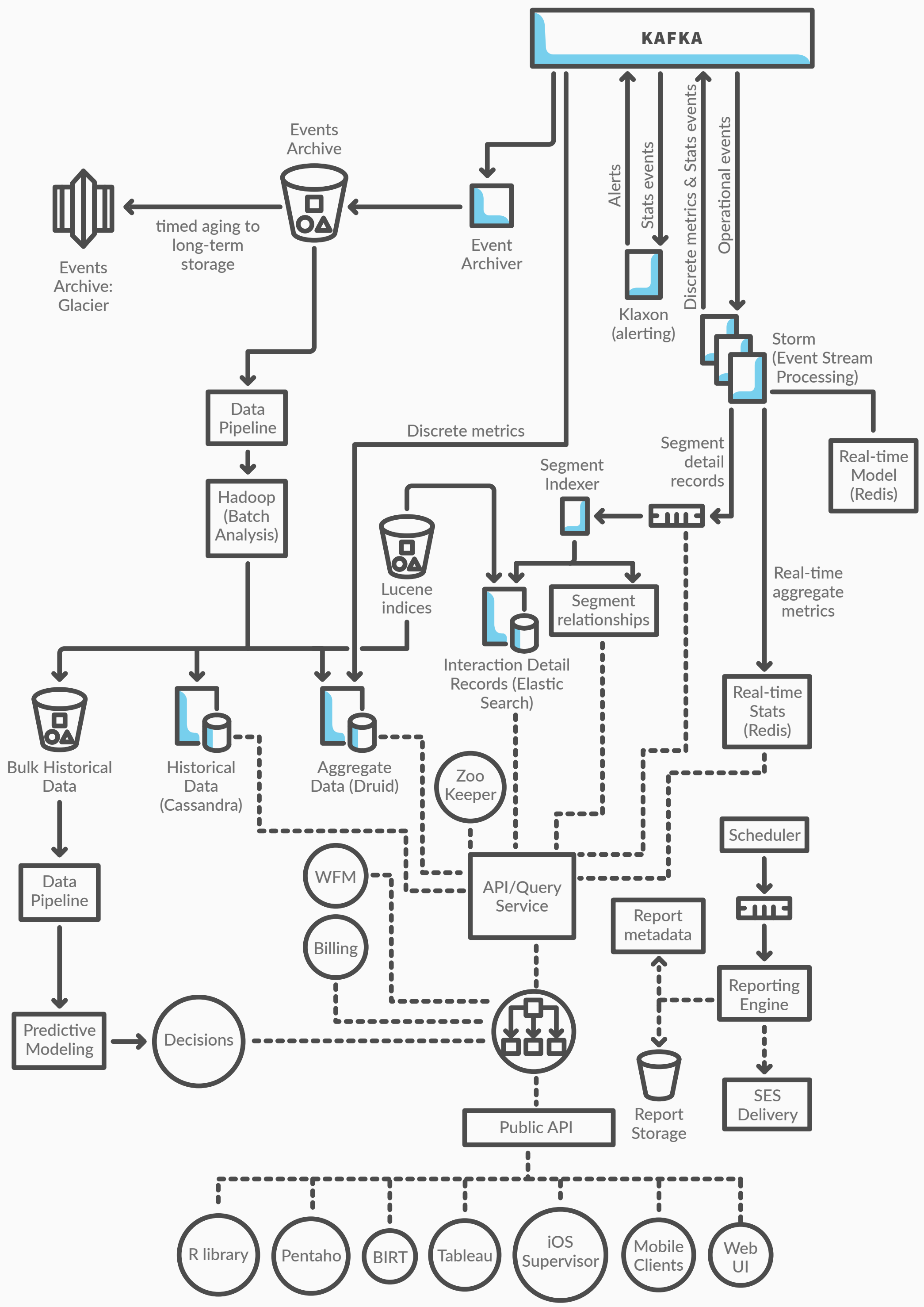


Fig 13.1 EVENTS QUEUES and REPORTING

Measure Everything!

Another advantage of scalable resources is that you can finally allocate for analyzing user and system behaviors properly, without affecting response times.

There are many tools to choose for analyzing data - for reporting issues automatically, and for checking reliability of requests.

Ongoing analysis is critical to a constantly scaling system with continuous deployment, and for understanding your customers and their trends.

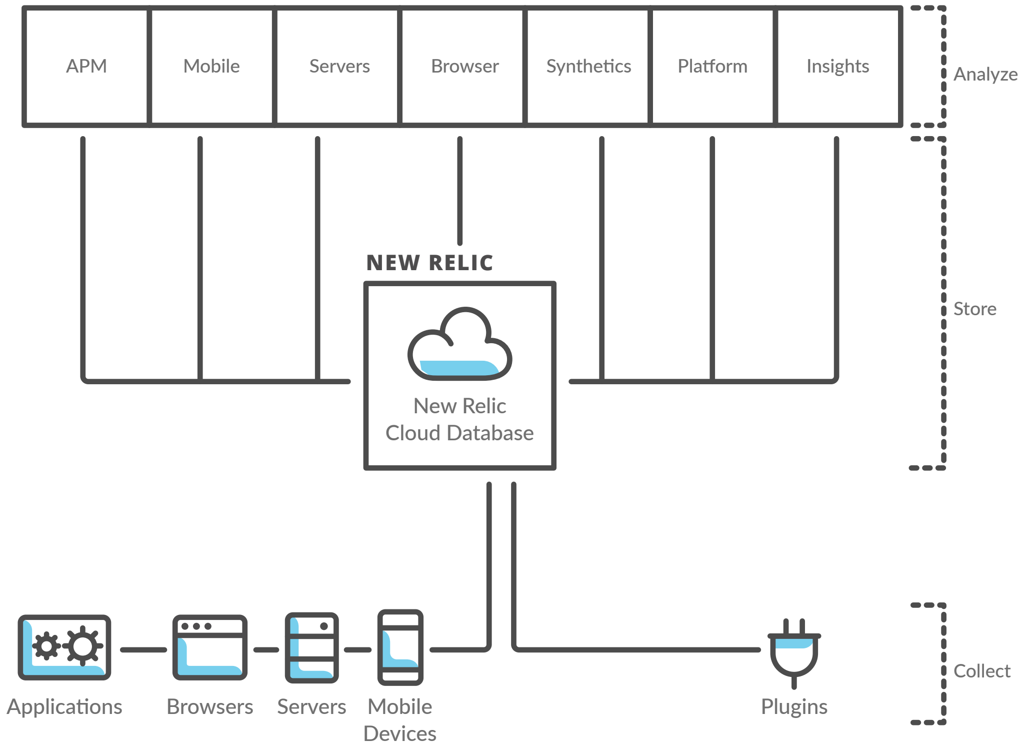


Fig 14.1 MONITORING for ANALYTICS

What is Interactive Intelligence PureCloud?

PureCloud has all of the features of the modern Cloud Application covered above, plus many more ideas under the covers. Agile continuous deployment accelerates feature development, directed by customers and the full history of Interactive Intelligence strategic contact center leadership. Each layer of PureCloud provides the foundation for another:

Platform provides the core cloud services

Collaborate adds management of organization data plus chat, document sharing, video chat, profiles, rules-based groups, search and a variety of ways to connect people

Communicate adds integrated telephony features to Collaborate to become a unified communications platform

Engage transforms Communicate into a comprehensive contact center solution

All features are fully integrated for a consistent unified experience.

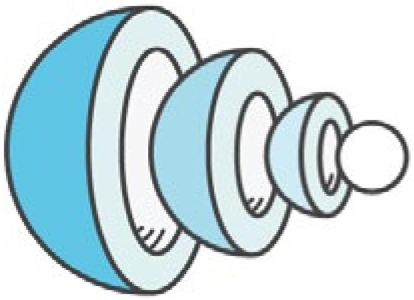
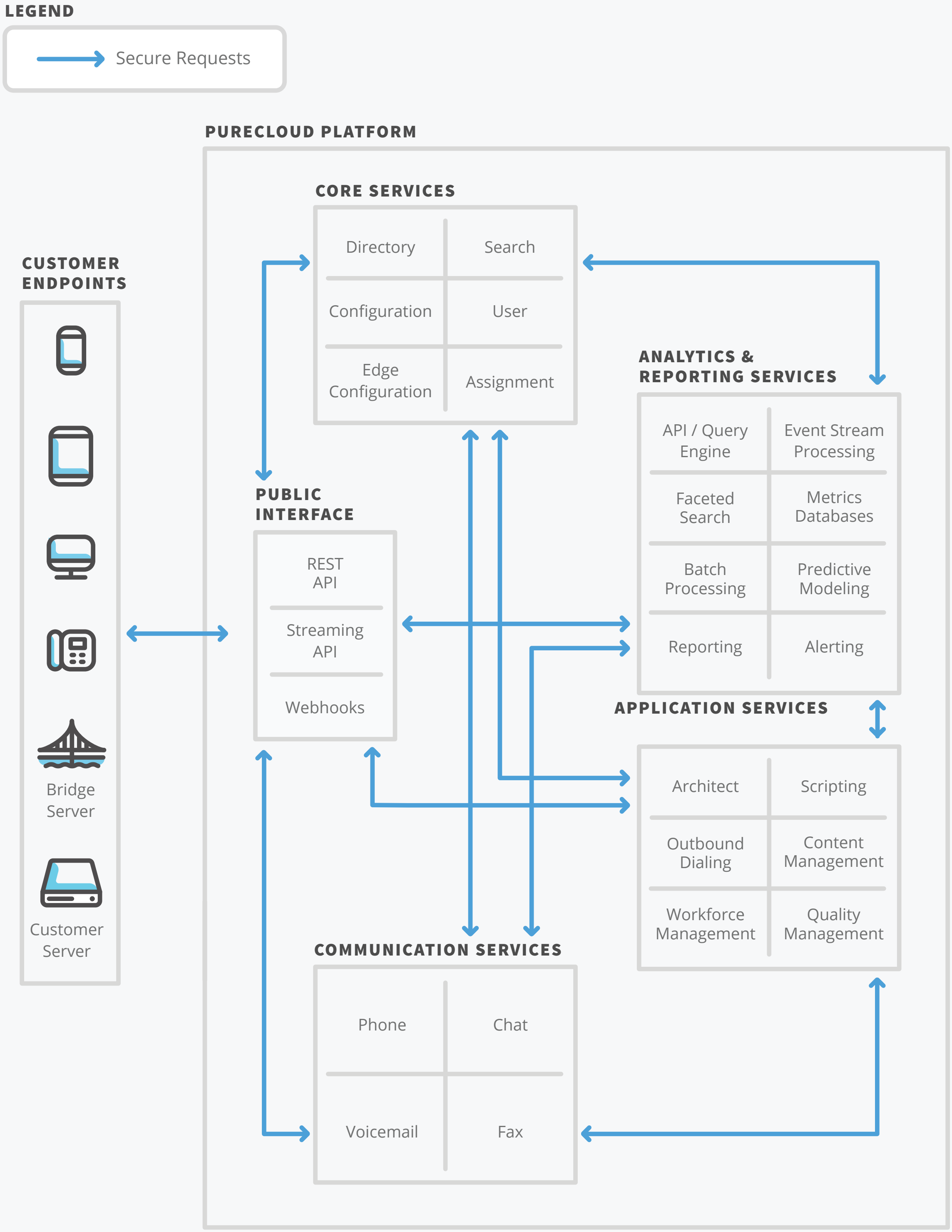


Fig 15.1 **LAYERS of PURECLOUD**

Each layer of PureCloud provides the foundation for another, with the platform at the center.

The PureCloud platform



Links

Amazon Partner Portal

AWS Cloud Compliance

Interactive Intelligence is an Amazon Web Services Advanced Technology Partner

Public API - PureCloud Developer Site

Get Started, REST APIs, Bridge Server Connectors, Webhooks, full online documentation

PureCloud Platform

More detail on the PureCloud Cloud Platform

PureCloud Open Source and GitHub

PureCloud uses Open Source code and actively supports the Open Source community, we also host example code for integrations on GitHub

PureCloud Data Integrations

Cloud-to-Cloud, Cloud-to-On-Premises, Webhooks, Data Dips, Screen Pops

Bridge On-Premises Data Connectors

Supported as of March 2016:
Microsoft Active Directory, Microsoft Exchange, Microsoft Sharepoint, Oracle Service Cloud, Workday, UltiPro, SalesForce, Interactive Intelligence CIC, Generic SQL Connector

About Interactive Intelligence

Interactive Intelligence Group Inc. (Nasdaq: ININ) provides software and cloud services for customer engagement, unified communications and collaboration to help businesses worldwide improve service, increase productivity and reduce costs. Backed by a 21-year history of industry firsts, 100-plus patent applications and more than 6,000 global customer deployments, Interactive offers customers fast return on investment, along with robust reliability and security. The company gives even the largest organizations an alternative to unproven solutions from start-ups and inflexible solutions from legacy vendors. Interactive has been among Software Magazine's Top 500 Global Software and Services Suppliers for 14 consecutive years, has received Frost & Sullivan's Company of the Year Award for five consecutive years, and is one of Mashable's 2014 Seven Best Tech Companies to Work For. The company is headquartered in Indianapolis, Indiana and has more than 2,000 employees worldwide. For more information, visit www.inin.com.

©2016 Interactive Intelligence. All rights reserved.

FIND US ON

ABOUT THE AUTHOR

Randolph Carter is a product designer and UX architect with broad history in user experience and enterprise application planning, software design and product development.

He holds CMMI, RUP, Omniture and ScrumMaster certifications. Randy has earned product design awards from BusinessWeek and other national publications, and has designed for Medtronic, Apple Computer, Peachtree Software, Citicorp, Chase Manhattan, Wachovia, Lulu, Bluestripe, and WebAssign.