# Game Build Distribution Tools to Speed Up Your CI/CD Pipeline

Quickly distribute large game builds to multiple locations and remote workers

## Introduction

Game development times are getting longer, and R&D costs are growing faster than ever before. The industry has reached the point where powerful new tools are needed to break these trends and enable publishers to release new titles to market faster.

Due to the increased size of builds and the sprawling scale of modern dispersed development teams, the challenge for game developers and DevOps has never been greater. They've reached the point where legacy systems and tools can no longer keep up to handle the content demands.

Now with Resilio's breakthrough capabilities for the movement of large files, over existing infrastructure, the delivery of builds to development, test facilities, and global partner sites can be streamlined, keeping release schedules on-track, while saving time and money. Once the delivered build is local, the Resilio solution accelerates distribution inside each facility, delivering near instantaneous and concurrent access regardless of team or build size and without adding additional local bandwidth and infrastructure.

## Challenges & Solutions Review

Game development is changing in ways that make distribution of builds increasingly difficult. It is common for multiple teams to participate in the development process to be located in remote locations. CI/CD practices mean that builds are distributed more frequently, stressing distribution systems and creating much greater data loads on key infrastructure. Finally, as the gaming experience becomes increasing richer, the size of the builds is growing by orders of magnitude.

To better understand the challenges modern DevOps teams are facing, it helps to first take a look at the existing solutions in common use, in the light of a real common scenario: Distributing Builds to 1000s of endpoints in multiple disperse locations. Then, at the end of this section, we'll explore how Resilio Connect overcomes all hurdles, to deliver the most reliable & fastest file distribution solution, to streamline daily CI/CD workflows.

### Simple File Transfer Solutions

#### Unreliable at Scale

At the atomic level, build distribution is simply moving large files from one location to another, over and over again. With this view, it is not surprising that early systems would be built using common and simple file transfer tools like FTP, scp, curl, wget and Robocopy.

These simple tools are far from state of the art and most eventually prove inadequate and are not easily scalable. All are based on TCP or rely on the client-server architecture. As a transport protocol, TCP can be challenged when distances between development facilities grows or when connecting networks are not the best quality, both eventually leading to slow distribution times. The client-server architecture suffers at scale and requires expensive infrastructure to overcome these limitations, or the result is again slow distribution times and limitations on concurrent access to reduce demand. Furthermore, these tools lack even basic data awareness and manageability functions, which means more work for DevOps to build and manage the scripts necessary to operate them.

All of these challenges are exacerbated when the number of development centers and the distance between them grows. Further pressure mounts when the size of the build and/or the size of the team that needs access grows. Unfortunately, all of these trends are unrelenting when it comes to modern game development, rendering these common tools obsolete.

## RSync

### Not Ideal For Large-Scale Distribution to Many Locations Over Long Distances

To help with manageability and the lack of data awareness in the simple tools above, many teams turn to rsync. With delta encoding, deduplication and the ability to detect changes and keep data in sync, this tool improves efficiency and manageability.

Rsync is a point-to-point solution and operates over TCP which inherently limits the ability to scale and with poor error recovery is not ideal large-scale distribution over vast distances and many locations common to game development.

## WAN Optimization Tools (to Overcome Long & Latent Networks)

### Performs Poorly at Scale & Costly to Maintain

To overcome latency and packet loss issues inherent with transferring data over long distances, some companies deploy WAN Accelerators in every facility. These products help with the TCP limitations in rsync and other simple file transfer tools and while expensive, are useful in point to point configurations between two facilities.

However, limitations are incurred when the number of facilities needing data grows beyond two. The capacity and effectiveness of the WAN Accelerator is effectively divided by the number of locations. If you have 10 facilities, the WAN Accelerator is only 10% as effective as a point to point configuration, since it has to repeat the same data transfer 10 times. They are also quite costly to maintain.

## DFS

### Suffers at Scale & Requires Expensive Investment In Infrastructure

DFS is a complex distributed file system across all locations to maintain a consistent and coherent view of all of the build data. While seamless consistency is a clear byproduct, these configurations also suffer at scale. Again, TCP and a lack of data awareness means the solutions are inefficient and suffer over distances without WAN Acceleration.

The complexity of configuration also means these solutions may not be ideal for all locations. Regardless of the above, when scale (meaning data or team size) increases, the centralized nature of the file servers in each location requires a vast and expensive amount of infrastructure to make the builds available locally.

## Open-Source P2P Solutions

### Complex to Build & Maintain, Lack of Deduplication & WAN Optimization Capabilities

To overcome scale limitations, some companies have turned to custom solutions using open source P2P implementations such as libtorrent which actually gets faster as demand increases.

However, these solutions are complex to implement and required developer resources to build and maintain. These resources are often scarce and best deployed against mission critical tasks in game development. Open source P2P lacks WAN acceleration for distant peer connections and data awareness and deduplication for basic efficiency. Also, the solution is typically difficult to manage, with no centralized view of P2P data flows and configurations.

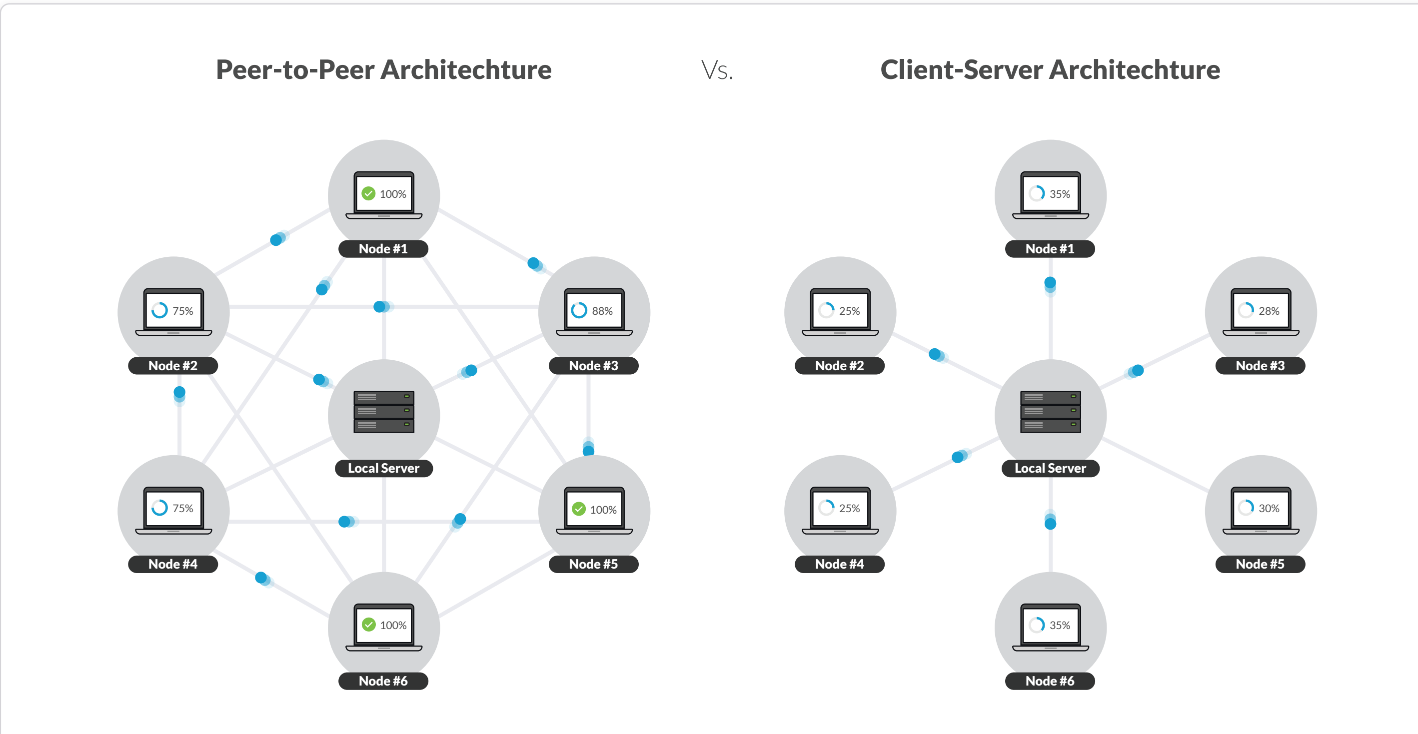
# Resilio Connect

## Finally, an Efficient, Reliable and Scalable Solution, Powered By Peer-to-Peer Technology

### Auto Scales with Your Needs

Resilio leverages P2P technology in ways that puts all of your infrastructure to work. As you can see in Figure 2 below, leveraging resources in all of the facilities allows the build to be distributed much faster and for less cost.

P2P turns every client into a server and gets dramatically faster as the demand increases. For large scale operations such as build distribution, it is the ideal technology to speed time to market when developing across many locations.




*[Diagram1: P2P vs Client-Server]*

### Intelligent Routing & Fault Tolerance

Resilio's namesake is "resilience" and combines P2P with intelligent routing to deliver the most resilient and fault tolerant system possible. For each facility and each connection, Resilio will find and use the fastest possible source of data intelligently. If the origin is down or not performing well for any reason, Resilio is able to draw on many other potential sources of data, delivering key infrastructure redundancy on the order of the number of nodes in the system. Even simple problems like a brief interruption of the connection will not require a complete restart of the distribution. Because of block level awareness, the Resilio solution seamlessly recovers and picks up right where it left off.

Resilio is a High Availability (HA) active/active system by nature. There is no operational reliance on any centralized systems, meaning that, once nodes and jobs are configured, they can operate semi-autonomously without any reliance on centralized systems, locations or the cloud. This reduces downtime and improves reliability.

### Efficiency & Data Awareness

The fastest data on your network is the data you never have to send. Because it's already there. Resilio incorporates block level data awareness and deduplication. If a block exists locally from an earlier build, the Resilio system will not request it. This reduces load on strained infrastructure while improving speeds significantly for many distribution jobs.

### Built-In WAN Optimization (Over µTP)

Resilio Connect combines WAN Optimization protocols for each P2P connection it makes, overcoming the point-to-point nature of legacy WAN Optimization alongside with the limitations of high packet loss and latency on the network. Furthermore, you don't need to deploy additional software or hardware or even maintain complex configurations for your WAN. Resilio seamlessly handles and optimizes every connection it makes.

### Pull & Push Strategy

Resilio Connect implements two ways of delivering builds to engineers. DevOps can push a build to a dedicated engineer or team. The destination user or team is defined via configuration from a management console (manually or via API). This is very useful as when the build system creates a build, it is automatically delivered to the designated team.

The pull method shows all available artifacts to a team member and that team member decides what to download. When the download is initiated, all data optimization protocols (WAN, P2P deduplication) work to deliver builds faster.

## Challenges & Solutions Summary

| | Simple File Transfer Solutions | RSync | Open Source P2P (Libtorrent) | DFS | Resilio (P2P) |
|---|---|---|---|---|---|
| Deduplication | — | ✓ | — | — | ✓ |
| Sync folders (delete propagation) | — | ✓ | — | ✓ | ✓ |
| WAN Optimization | *Requires Additional Software / Hardware* | | | | ✓ |
| Scalable | — | — | ✓ | — | ✓ |
| Extensive error recovery | — | — | — | — | ✓ |
| File integrity verification | — | ✓ | ✓ | ✓ | ✓ |
| Pull and push strategy | — | — | — | — | ✓ |
| Smart routing | — | — | — | — | ✓ |
| Encrypted data transfers | — | — | — | — | ✓ |

# Workflow Examples

## Example #1: Build Distribution From an Artifactory Server to Remote R&D Servers & Endpoints Over WAN & LAN

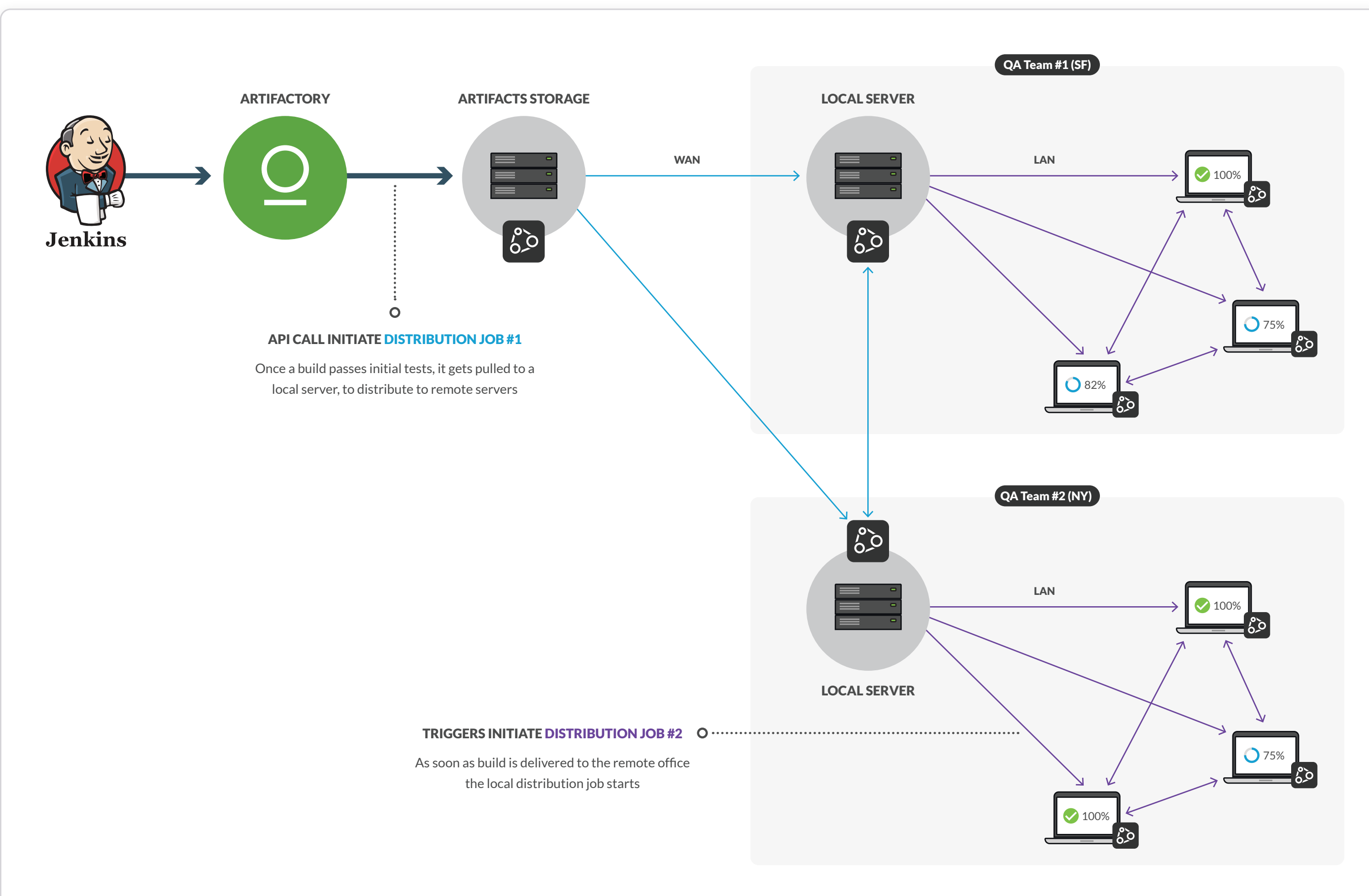
This example shows a staged distribution of builds. The first stage, Job #1 distributes builds to remote offices. Depending on the build it may or may not be distributed to local teams. If the builds need to be distributed to teams, Job #2 performs local distribution of the build as shown in the figure below.

### Steps:

1. The build system (Jenkins/Travis as an example) builds a new game build
2. It stores binary in the Artifactory/GFS or other build storage systems
3. If the build passes initial tests and ready for distribution build is pulled to a server to distribute to remote teams
   - Connect Agent start **Job #1 (blue line)** to distribute builds to servers in the remote offices
4. As soon as build is delivered to the remote servers, the local distribution job starts
   - The **Job #2 (purple line)** pulls build from the local server and distributes it to all team members

### Notes:

Job #1 is accelerated and scalable to an unlimited number of offices while Job #2 ensures the local professionals in each office are not left waiting for access to the local build server.




*[Diagram3: Workflow Example #1]*

# Example #2: Build Distribution From an Artifactory Server to Remote R&D Endpoints Over WAN & LAN (Without Using Local Server In Remote Locations)

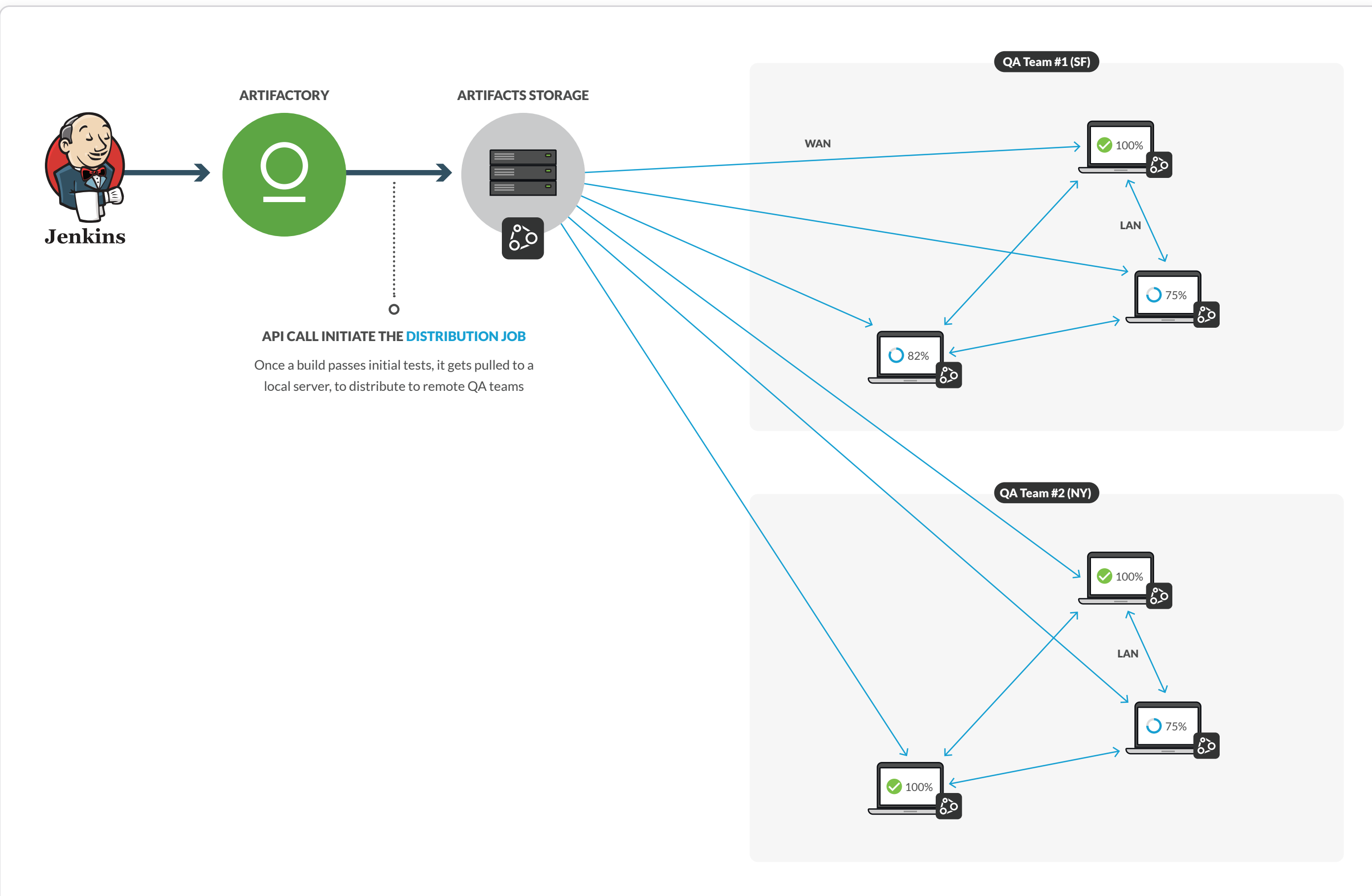
This example shows distribution of builds without a server in the remote location. This architecture is extremely important for small offices where you can't put dedicated servers or overs the case where teams are mobile or working remotely. There is a single distribution Job that delivers builds from the build server directly to developer's machines without need to use a server in the remote offices. This approach simplifies infrastructure since less hardware is involved. Each machine downloads data first from local devices and only if there are no other pieces available, it pulls it from central server.

## Steps:

1. The build system (Jenkins/Travis as an example) builds a new game build
2. It stores binary in the Artifactory/GFS or other build storage systems
3. If the build passes initial tests and ready for distribution build is pulled to a server to distribute to remote teams
   - Connect Agent start the **Job (blue line)** to distribute builds to machines in all remote offices (over WAN & LAN)

## Notes:

This example illustrates how Resilio Connect can help DevOps dramatically reduce infrastructure costs while also delivering builds faster than ever, even to remote development teams. It also illustrates how Resilio is able to overcome a wide variety of network conditions to reliably and effectively distribute builds even over bad networks.




*[Diagram3: Workflow Example #2]*

## Example #3: Route Data & Builds to Teams Through a Series of Policies & Job Configurations

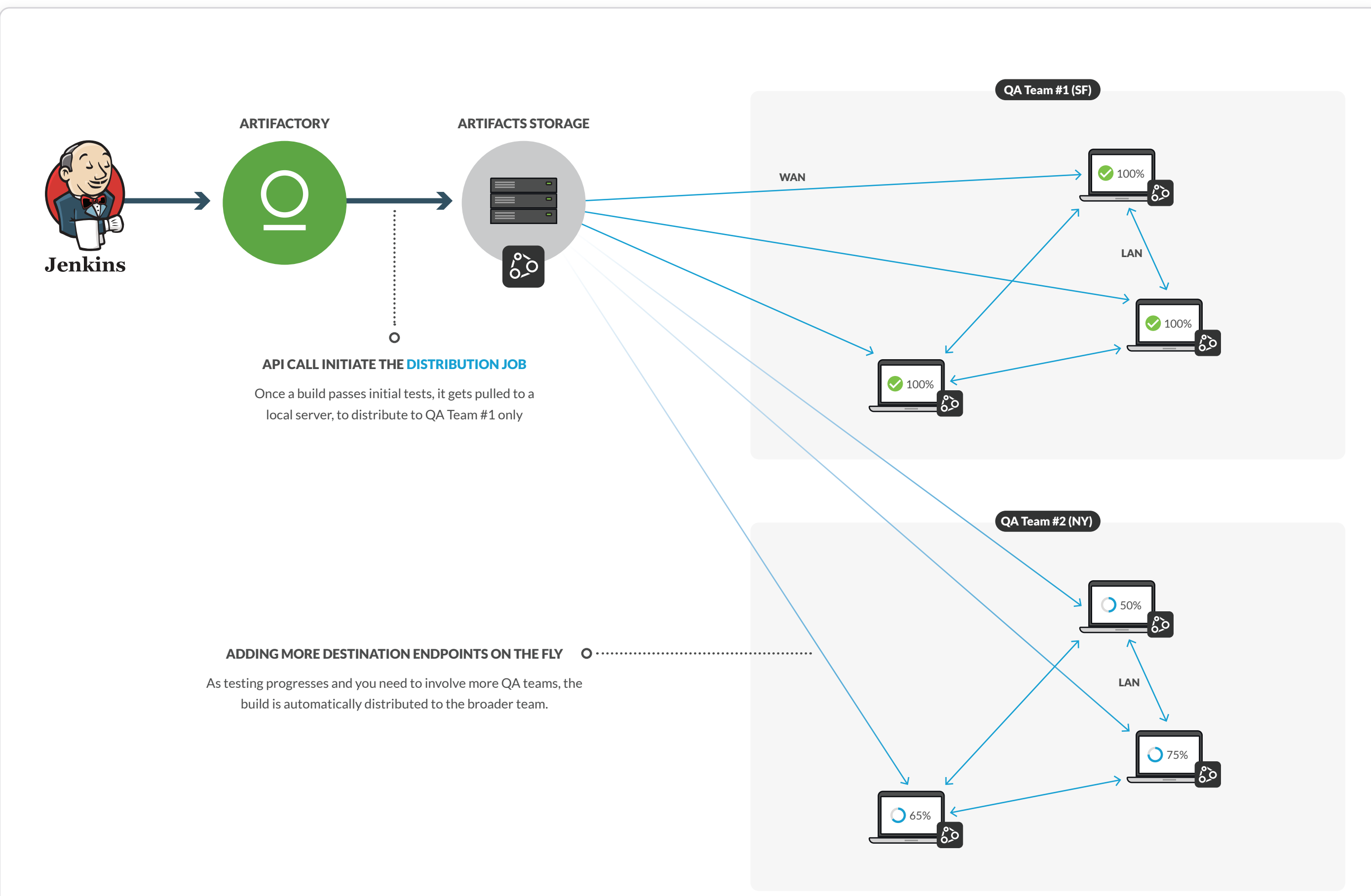
This example covers a case when you start testing a build with small team that could be located in one or several offices. As testing progresses and you need to involve bigger teams, the build is automatically distributed to the broader team.

### Steps:

1. The build system (Jenkins as an example) builds a new game build
2. It stores binary in the Artifactory/GFS or other build storage systems
3. If the build passes initial tests and ready for distribution build is pulled to a server to distribute to remote teams
   - Connect Agents start the **Job (blue line)** to distribute builds to machines of the initial testing team
   - As build progress thru the process the second team in another office needs to be involved **(gradient lines)**. The designated agents added to distribution job, and Connect start transferring the data

### Notes:

This example illustrates the high degree of control that Resilio offers to DevOps teams, with an ability to programmatically and intelligently route data and builds to teams through a series of policies and job descriptions. These policies can be set through the central management console or programmatically through the API.




*[Diagram3: Workflow Example #2]*

# Key Benefits

## Improve Speed and Server Load While Lowering Your IT Infrastructure Costs

By far the biggest issue is distributing a large build from a central server to developers and QA machines. The computers involved are very powerful and the network is fast and consistent such that even small amount of these machines could generate a load that central server can't support. If a facility uses SMB/NFS to access these servers, even a simple copy requires the transfer to start over if the connection is interrupted. If developers instead use tools like ftp/scp the problem is the load on the central servers.

At this point, there are two choices facing classical DevOps systems (neither represents an ideal solution):

1. Deploy a massive amount of local server, storage and network infrastructure (capital intensive)
2. Rate limit the number of developers and QA professionals who can access the build concurrently (labor intensive, or more accurately, labor inefficient).

**A far better approach is to leverage infrastructure that already exists to assist with this task, namely the powerful QA and developer machines and gaming consoles that mostly sit idle otherwise.** The Resilio Connect Agent is a small piece of P2P software easily deployed on these machines that can put them to work, speeding the delivery of every build to neighbors on the LAN instead of relying exclusively on centralized servers. In this sense, the more developers you have, the faster the Resilio solution gets. And what's more, using these otherwise idle computing resources can reduce or eliminate the need for costly centralized server infrastructure.

In this sense, the ROI on Resilio is very straightforward when compared to the extensive cost of upgrading and overhauling local servers, storage and network infrastructure in every developer facility.

## Easily Integrate Resilio Connect Into Existing Workflows By Using Triggers & Connect's REST API

To ease integration into existing systems, Resilio Connect has the concept of triggers. Triggers are specific events that fires script execution at the beginning of transfer, after transfer is completed or after all destination users have received the build. This allows various integrations on QA machines, such as updating builds on the machine, showing notifications to the user that the new build is available or any other related actions.

The Resilio Management Console has a REST API to control artifact distribution. The API uses command channel to define groups of users, policy, schedule build deliveries and control other various job transfer parameters. The REST API allows delivery system to easily integrate into any existing workflow as well as report progress on existing data transfer.

# Summary

Scalability challenges for build distribution can manifest in many ways. Even distribution across a local network within each office can suffer if the local infrastructure is not sufficient to meet the demand. Resilio offers a unique software and capabilities to break the cycle of never ending infrastructure upgrades in each office, delivering a cost effective and highly scalable system for local build distribution that is faster than anything you have ever seen.