**Manage Agile software delivery plans across teams**

**Introduction**

* 1 minute

Azure DevOps can help your team scale their development processes to deliver bigger and more ambitious projects. Let's rejoin the Tailspin Toys team as they tackle a common problem organizations face as they grow. Together, we'll discover what it takes to effectively manage work schedules across multiple teams.

You met the team at Tailspin Toys earlier in this learning path. In the previous module , you watched as they began to manage their work schedule through Azure Boards. Soon, word of their early success began to spread, and now other teams are exploring how they can enjoy the same benefits.

As more teams adopt Azure Boards, the organization begins to see the network effects of having their planning consolidate around a consistent process. Problems that were once written off as a cost of doing business now have achievable solutions. Let's watch the team as they lead their organization in its evolution.

In this module, you will:

* Learn how *delivery plans* enable multiple teams to plan, schedule, and coordinate their work.
* Install the Delivery Plans extension for your Azure DevOps project.
* Create a delivery plan and adjust a team's sprint workload to optimize delivery efficiency.

**Prerequisites**

The modules in this learning path form a progression. We recommend you start at the beginning of the Evolve your DevOps practices  learning path before you work on this module.

If you'd rather do only this module, go through Get started with Azure DevOps  first to set up your Azure DevOps organization.

**Meet the team**

You met the *Space Game* web team at Tailspin Toys in previous modules. As a refresher, here's who you'll work with in this module.



Andy is the development lead.



Mara just joined as a developer and reports to Andy.

Mara has prior experience with DevOps and is helping the team adopt a more streamlined process by using Azure DevOps.

# What are delivery plans?

* 6 minutes

As development organizations grow, they need to reorganize into smaller teams that can efficiently manage portioned units of work. These teams will usually have their own work schedules, boards, and other processes that meet their unique needs within the context of the larger goals of the organization. Over time, organizations may find that they enjoy network benefits by consolidating their processes around a consistent framework.

A delivery plan is a visualization of one or more work schedules. It is intended to provide teams and management an overall view of what each team is planning to produce and when. This allows decisions to be made that optimize the investments across the organization.

It's important that teams regularly review their delivery plans in order to make sure that their work schedule aligns with the schedules of other teams. These reviews should ask questions like:

* Are we sure we can deliver what we have committed to on our current schedule?
* Are we confident that the teams we depend on will deliver what we need on their current schedule?
* Are there lulls in our schedule that we could fill with work?

Delivery plans add value at any point in a project's lifecycle. Since they are dynamically generated based on team backlogs, they're always up-to-date and offer the latest insights.

Let's join the Tailspin web team in their discussion.

**Andy:** I just had a great meeting with engineering management. I demoed the work we're doing with Azure Boards and they're excited about the prospect of getting other teams on board.

**Mara:** Awesome! When will they get started?

**Andy:** That's the best part—they already have! Last night the game engine project lead created a team with some sprints and began adding work items. I took a quick look this morning and it's shaping up nicely. Let me show you what they're up to.

Andy navigates to the game engine's current sprint board. He and Mara review the work items with great interest.

**Andy:** Hmm...I just noticed that they're not planning to deploy their beta by the end of this sprint. Aren't we expecting to integrate our leaderboard with the beta database during our next sprint? We can't do that if they don't ship the beta first.

**Mara:** That's a good point. We have a dependency on that team to produce that deliverable so that we can produce one of our own.

**Andy:** This could have really hurt our productivity next sprint. I'm going to give them a call to find out what's going on.

Unfortunately, more sophisticated team structures can result in gaps or lags in communication. When one team is blocked, they might not be able to produce something another team is dependent on. This might not be a major issue for a small group of teams that have daily meetings for all concerned. However, as teams scale in size and location, it can become untenable for everyone to know everything going on everywhere. It's at this point where organizations need to transition from a pure "push" model (like in-person or email announcements) to a "pull" model (where teams can review and track each other's schedules).

**Andy:** Okay, I just spoke with the dev lead. She told me that their team is blocked on shipping the beta until the art team returns from Cliffchella.

**Mara:** The mountaintop music festival?

**Andy:** Yeah, apparently it's a huge deal in the design community and their entire team just drops off the grid for a whole week to attend. The game engine team is pretty upset because it slipped their schedule by three weeks. Had they known it was coming, they would have made sure to get the artifacts they needed ahead of time. They also apologized for not letting us know sooner. They didn't realize we would be waiting on their beta to ship our part.

**Mara:** Well at least we can be glad that the game engine team is publishing their sprint plans. It helped us find this dependency issue early enough to adjust our schedule.

**Andy:** I just wish there was a way to see these potential risks coming more easily. Our teams have so many dependencies across the company that there's no way we can attend every meeting and subscribe to every distribution group.

**Mara:** We should create a delivery plan so that we can see our team sprints side-by-side. This will help both teams more easily identify how our schedules impact each other. And I know the perfect Azure DevOps extension for the job.

## Recommendations for managing multiple Agile teams

An Agile approach, along with Azure DevOps, can substantially improve project transparency and predictability. However, projects may still run into traditional challenges, often related to personnel or miscommunication. Here are a few things to consider as you scale your Agile efforts.

### Build trust in your people and processes

Early detractors from Agile implementations are often skeptical about their ability to improve team performance. It's important for thought leaders within the organization to build trust by illustrating how the tools and processes produce results. Sometimes these results are improvements in productivity, and those are easy to quantify. However, don't forget to highlight the team wins that occur by circumventing potential problems, such as avoidable schedule slips or quality issues. As people begin to associate the benefits with the process that achieved them, you will get more enthusiasm.

### Elevate the organization above the team (and individual)

Some teams and individuals get territorial when new processes or policies are proposed. Rather than framing new policies as negatively exposing the performance of specific teams or individuals, highlight how the new transparency across the organization informs everyone of expectations. Having a single place where anyone can trace how their work relates to the organization meeting its goals will drive home the importance of their commitment to the process.

### Foster a culture of transparency

Unfortunately, the term "transparency" gets a bad wrap. Nobody asks for more transparency when everything is going great. Instead, transparency (or lack thereof) is often blamed when teams are struggling. Even with all of the opportunities for transparency afforded for Agile teams, it's still subject to the honesty of individuals and teams. Emphasize that one of the reasons for transparency is to be able to identify and address potential issues before it's too late. Everyone understands that people sometimes run into circumstances that prevent them from meeting schedule deadlines. But if they don't feel safe in reporting disappointing news until the last possible moment, it can have a much more destructive impact. Building a comfort level with transparency can start with thanking people for reporting expected delays as early as possible.

# What is Delivery Plans?

* 6 minutes

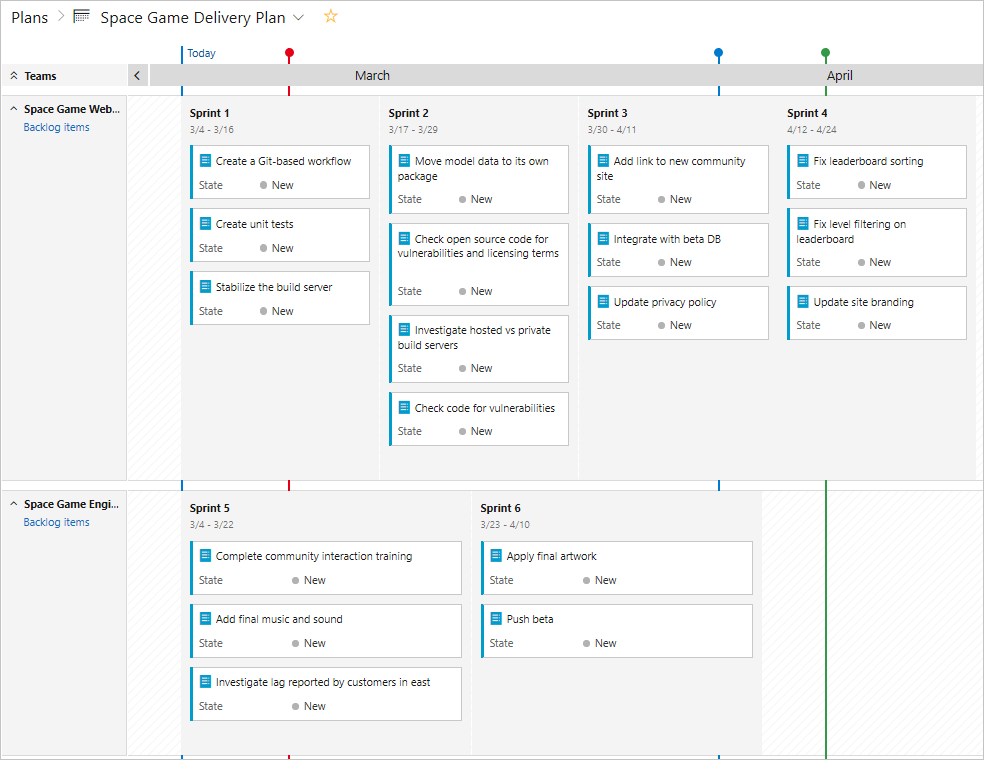
Delivery Plans is an extension for Azure DevOps that helps organizations plan and review work schedules across multiple teams. The Tailspin team will use this extension to get a better idea of how their work relates to work to be produced by other teams.

Mara installed Delivery Plans in the team's Azure DevOps organization. She then created a delivery plan and added the sprints for her team and the game engine team. Excited to show off the potential, she invites Andy over for a quick demo.

**Mara:** After our last conversation I looked into our options for managing delivery plans. I found an Azure DevOps extension that seems to give us everything we need.

**Andy:** I'm very interested to see what you've come up with. There's a lot of stress throughout the organization about the beta slip, so anything we can do to improve schedule efficiency will be quite welcome.

**Mara:** Okay, here it is.



**Mara:** The Delivery Plans extensions allows us to create a "delivery plan". Once we create that, we can add in the backlogs of teams within the organization. They are shown in parallel so that we can see what each team is planning to deliver against a calendar backdrop.

**Andy:** This is great! Now we know when something we're dependent on won't be available in time. We can even gauge the likelihood of delay based on how much other work and dependencies those teams have taken on. This should help mitigate some of the "schedule chicken" behavior that sometimes go on around here.

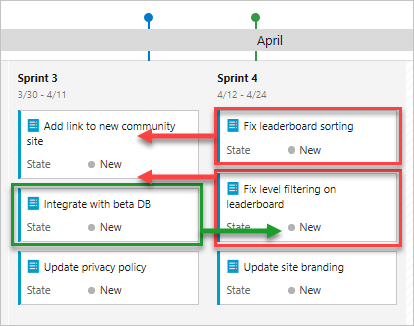
**Note**

**Schedule chicken** is when two or more teams are at risk of not meeting deadlines, but none of them want to admit it. Instead, each wait for another to slip their schedule first and then use the other team's slip as a pretext for delaying their own delivery.

**Mara:** Yes, and we can also use this as an opportunity to let other teams know if we're going to slip something they're dependent on. It will help us build trust in our people and processes.

Andy nods in agreement. It would be nice for the teams to have more faith in each other.

**Andy:** Well now that we know about the beta slip we have to move our associated work out to a future sprint. On the bright side, it gives us an opportunity to pull some new work in to replace it. Let's swap the integration work with those two leaderboard bugs. Mara drags the integration work item out to the following sprint. She then drags the two leaderboard bugs back in to fill the available capacity.



**Mara:** I also added the current beta date as a milestone. Now we will always have it in place as a reference point for the work we're planning.

**Andy:** We should also add events like Cliffchella and the annual company party.

**Mara:** Why the company party? Does that impact the schedule?

**Andy:** It might. Every year the DBAs enter the pie-eating contest and they all end up calling in sick the next day. I'm not saying we should expect it to happen again this year, but I do think we should be prepared. And now we have the tools for it.