Internal Agile Delivery Model

Agile Delivery Model

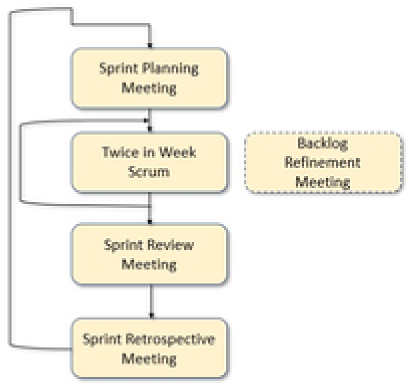
Current DevOps and cloud team used Jira to manage sprint board using following Hierarchy levels.

- **Initiative** - Drive organization directions and measurable efforts on quarterly basis
- **Epic** - An epic is a large body of work that can be broken down into a number of smaller stories
- **Stories** - Stories are time bound efforts to achieve sprint goals and add values to each sprint

We have to make sure your measured effort logged as stories and align within epic or initiative efforts. Apply fix version (release date) for each story to generate Release Notes and understand team progress by monthly
We use portfolio plug in to assign different themes which can help senior managers to monitor team efforts
There are five themes, let's make sure we used these five categories for each story we work in each sprint.

1. **BAU** - Standard internal team building/mentoring/Sprints
2. **Level Up** - Increase skills/knowledge in DevOps/Agile/Cloud area doing training/certifications
3. **Differentiators** - Build product/services to provide edge for DevOps and Cloud solutions.
4. **Business Growth** - Work involved in client workshops/assessment / POC / Initial kick off
5. **Efficiency** - Automation, infrastructure as code, metric and reporting

A sprint is a short, time-boxed period when a scrum team works to complete a set amount of work. Sprints are at the very heart of scrum and agile methodologies, and getting sprints right will help your agile team ship better software with fewer headaches. "With Scrum, a product is built in a series of iterations called sprints that break down big, complex projects into bite-sized pieces,"
DevOps and Cloud team have two-week sprint in place during which the team is developing different capabilities for engagement practices. Each sprint starts with a planning meeting where the Product owner presents their Product backlog for discussion. The team then decides how much of the requested functionality they will be implementing in this sprint. They also decide how to break work down into specific actions that need to be taken. During the sprint, the team holds bi-weekly Scrums to ensure synchronization.
At the end of the sprint, a Review is done with the product owner to achieve a common understanding of what has been achieved and whether the set goals were reached. It is followed by a Retrospective during which the team determines action items to improve their work and prevent experienced problems from reoccurring in the future.



Scrum Flow

Estimating Agile Projects with Story Points

One benefit of Scrum and other Agile practices is predictability. Predictable Agile teams know their capacity for work, make realistic commitments, and deliver what is expected. But predictability requires good estimation. Good estimates enable an Agile team to plan sprints well, forecast milestones, and build confidence with end users and other stakeholders.
An Agile team must estimate product backlog items (e.g., stories) accurately to know its capacity for work. Without consistent and accurate estimates, the team's past accomplishments cannot serve as a basis for velocity, which helps the team commit to whatever it can reasonably accomplish in the future. According to the Scrum Guide™, "Only what has happened may be used for forward-looking decision-making."
We prefer using points to estimate stories rather than hours. Story points are approximate and relative estimates that a team makes together about the effort required to complete a story. Points should reflect the amount of work involved, its complexity, and any uncertainties. The team should base its story point estimates on reference stories it has completed in the past. Once team members agree on the point values of reference stories, they then assign values to new stories by comparing them with the references.
Few best practices for estimating story points include:

1. Use non-consecutive point values. Sufficiently separate the values to make each estimate more clear-cut. Popular patterns are Fibonacci series (1, 2, 3, 5, 8, 13, 21, etc.) and powers of 2 (1, 2, 4, 8, 16, etc.). Anything estimated above a 16 should be broken into multiple stories and re-estimated.
2. Stabilize the team. Don't make important decisions using estimates calculated during the early forming stages of a team's evolution or while members are coming and going. Until a team's membership, estimation process, and velocity stabilize, the estimates will not be accurate.
3. Get inputs from everyone (business-layer developers, front-end developers, database developers, testers, etc.). Only the development

team (not the Product Owner) can estimate the points but it is important that estimates reflect the perspective of everyone involved in completing the story.

4. Make informed estimates. The Product Owner should present the story and acceptance criteria to the development team and answer questions during an interactive backlog refinement meeting. The team should capture any details they need to remember when implementing the story.
5. Iterate and improve. Review the team's performance during retrospectives and adjust as needed. Keep the reference stories current and recalibrate stories on the backlog regularly. You can guard against "point inflation" by comparing new estimates with stories estimated in the past.
6. Don't overthink it. The team should arrive at a consensus on the points for a story in a reasonable time. Don't try to convert hours to points or vice versa when estimating. However, once story points are assigned, let development team members estimate their tasks in hours.

While the team ultimately decides how to estimate, applying these best practices will help it accurately estimate the amount of work completed in a typical sprint as well as the work left to do. Once the team knows those values, it can accurately estimate the number of sprints or iterations needed to complete the backlog items assigned to a release. And knowing the number of sprints (with the sprint duration) will enable the team to predict release dates and other milestones. Knowing the point value estimates of stories or groups of stories can also help the Product Owner prioritize items on the backlog and determine Return on Investment (ROI).

Sprint Ceremonies

Some of the more well-known components of the Scrum framework are the set of sequential events, ceremonies or meetings that scrum teams perform on a regular basis. The ceremonies are where we see the most variations for teams. For example, some teams find doing all of these ceremonies cumbersome and repetitive, while others use them as a necessary check in. Our advice is to start out using all of the ceremonies for two sprints and see how it feels.
Below is a list of all the key ceremonies a scrum team might partake in
Organize the backlog:
Sometimes known as **backlog grooming**, this event is the responsibility of the product owner. The product owner's main jobs are to drive the product towards its product vision and have a constant pulse on the market and the customer. Therefore,
he/she maintains this list using feedback from users and the development team to help prioritize and keep the list clean and ready to be worked on at any given time. A well-prioritized agile backlog not only makes release and iteration planning easier, it broadcasts all the things your team intends to spend time on.
Successful Agile teams manage their backlogs (prioritized queues of work) well. A team typically maintains backlogs for the product being built, incremental releases, and each iteration. According to the Scrum Guide, "The Product Backlog is an ordered list of everything that might be needed in the product and is the single source of requirements for any changes to be made to the product. The
Product Owner is responsible for the Product Backlog, including its content, availability, and ordering." The Product Backlog is a "parts list" of user stories, bug fixes, epics, and tasks that describe features, functions, enhancements, and fixes for current and future releases. It's what the development team converts into a working product in increments called sprints or iterations.
Creating the backlog is not easy. Requirements collection and specification is largely an art and it takes time to fully define the vision for a product. User stories that describe that vision should be Independent, Negotiable, Valuable, Estimable, Small, and Testable (INVEST).
Each user story or epic should describe a specific benefit. Product backlog items should include a priority, effort estimate, and some quantification of organizational value. The backlog should be visible to everyone as well as detailed, estimated, emergent, and prioritized (DEEP). Once a product backlog exists, dynamics inside and outside an organization make it a challenge to manage. As an organization's internal and external environments change, the product vision (and hence, the backlog) must evolve with them.
The Product Owner should involve other stakeholders (development team members, business representatives, users, etc.) when maintaining the alignment. Backlog refinement (i.e., grooming) manages the vision by collaboratively adding details, estimates, and priorities to backlog items with the goal that two iterations of backlog items are always "ready" for development. That means the items are stable; can be completed by one person within a sprint; have acceptance criteria, priorities, and effort estimates; and are impediment-free. Typical grooming activities include removing irrelevant user stories, creating new stories, reassessing priorities (especially after adding new stories), adding/revising effort estimates (preferably with story points!), and splitting user stories into smaller units of work.
Here are best practices to improve your product backlog management

1. Have a single Product Owner. If you have multiple people in this role (e.g., because there are multiple decision makers or multiple Scrum teams), then designate one to be the lead to provide consistency and unity.
2. Consider personas (user types) and workflows (e.g., typical day or interaction session) to identify all the user stories needed to complete a product vision. Also consider exceptions, user interface details, technical implementation, and sub steps when writing user stories.
3. Include stories that address technical debt. Make sure the development team, Product Owner, and other business stakeholders agree on the percentage of time to spend on technical debt. Over time, I recommend the team reserve 20-40% of its capacity to pay off technical debt, refactor, and explore architectural innovations. This leaves 60-80% of the team's capacity for new features.
4. Quantify the business value of backlog items and use that to drive prioritization.
5. Also consider dependencies and risk when prioritizing backlog items. Note dependent tasks that need to be coordinated with other teams and complete high-risk features sooner.
6. Maintain a release plan, which maps backlog items to incremental releases. Ideally, the release plan will reflect target markets and expected outcomes.
7. Use a backlog planning/management tool to capture items and their attributes (e.g., priorities and estimates), sort items by priorities, flag impediments and dependencies, and facilitate sprint planning by assigning the items to an iteration.
8. Each team should spend 1-2 hours per week on backlog refinement. Don't wait for a sprint to end to start preparing the next set of items because team members will be focused on completing their assignments and the process might be rushed.
9. Remind team members one business day in advance of backlog grooming sessions. Do informal backlog grooming with a smaller group before larger, more formal sessions
10. Make a conscious decision to add something to the backlog. It's okay to throw an epic on the backlog to capture something during a brainstorming session but understand the cost to keep items on the backlog. Once they are there, items need to be reviewed and refined,

which takes time away from other potentially more valuable tasks.

Sprint planning:

The work to be performed (scope) during the current sprint is planned during this meeting by the entire development team. This meeting is led by the scrum master and is where the team decides on the sprint goal. Specific use stories are then added to the sprint from the product backlog. These stories always align with the goal and are also agreed upon by the scrum team to be feasible to implement during the sprint.
At the end of the planning meeting, every scrum member needs to be clear on what can be delivered in the sprint and how the increment can be delivered.
Goal of sprint planning is to focus execution, minimize surprises, and guarantee overall higher quality code
Sprint planning involves two key tasks:

1. Grooming the backlog:

a. Prioritizes each work item, with the most important work with priority (high, low, medium)
b. Includes fully-formed user stories with acceptance criteria for agile team
c. Contains an up-to-date estimate for each work item

1. Deciding which work to complete in the upcoming sprint.

At the end of sprint planning its good practice to get verbal approval from everyone about what the team is actually committing to shipping at the end of the sprint.

Sprint:

A sprint is the actual time period when the scrum team works together to finish an increment. Two weeks is a pretty typical length for a sprint, though some teams find a week to be easier to scope or a month to be easier to deliver a valuable increment.

1. Create the Product Vision & Product Backlog Together

Often the customer has an idea about the product and its possibilities but a Product Backlog is still missing. A good practice is to organize a workshop with the customer and the supplier's Development Team and letting them create the Product Backlog together. Even before setting up the contract. Ideally you also create the product vision to ensure mutual understanding. By creating the product vision and Product Backlog together the customer and supplier really get to know each other. More important: the customer meets the actual Development Team. They are the ones who are going to realize his dream. They are the ones he needs to trust.

2. Estimate the Implementation Effort of the Product Backlog Together

After creating the Product Backlog together it's also possible to estimate it in the presence of the customer. The advantage is the customer hears all the discussions & questions and can answer them directly. Possible complexity is also detected and discussed together which ensures mutual understanding about the given estimations.

3. Determine the Business Value of the Product Backlog Together

When the Product Backlog is estimated in implementation effort, the next step is determining the business value. This is up to the stakeholders. In the same session, facilitate the stakeholders to discuss and determine the business value for every Product Backlog Item using business value points. The goal is a shared understanding of the priorities and inviting the stakeholders to participate in defining what is more valuable. This exercise forces them to prioritize the Product Backlog without delegating this responsibility entirely to the Product Owner.

4. Clarify Dependencies Between Product Backlog Items

When the implementation effort and business value is determined for every Product Backlog Item, it's time to detect the dependencies. Technical dependencies can be clarified by the Development Team, functional dependencies might also be detected by the Product Owner and stakeholders.

5. Start Small

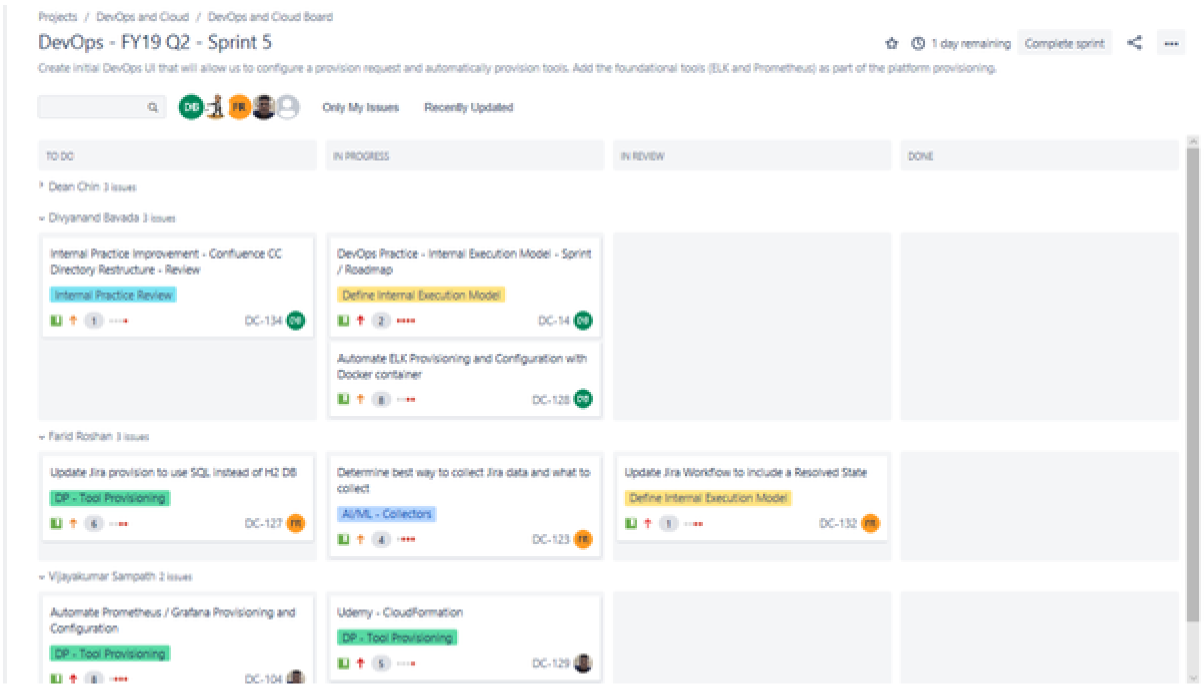
Even when a customer has a huge budget for his project, first agree upon doing only one Sprint. After you've created and estimated the product vision and Product Backlog together, only do the first Sprint with the goal of delivering the first 'done', valuable and potentially releasable increment. Perform a Sprint Review and Sprint Retrospective and decide if there's enough mutual trust to start another Sprint.
All the events — from planning to retrospective — happen during the sprint. Once a certain time interval for a sprint is established, it has to remain consistent throughout the development period. This helps the team learn from past experiences and apply that insight to future sprints.

Daily/Alternate Day Scrum or Stand Up:

This is a daily super-short meeting that happens at the same time (usually mornings) and place to keep it simple. Many teams try to complete the meeting in 15 minutes, but that's just a guideline. This meeting is also called a 'daily stand-up' emphasizing that it needs to be a quick one. The

goal of the daily scrum is for everyone on the team to be on the same page, aligned with the sprint goal, and to get a plan out for the next 48 hours.
The stand-up is the time to voice any concerns you have with meeting the sprint goal or any blockers. A common way to conduct a stand up is for every team member to answers three questions in the context of achieving the sprint goal:
• What did I do yesterday? • What do I plan to do today? • Are there any obstacles?
However, we've seen the meeting quickly turn into people reading from their calendars from yesterday and for the next day. The theory behind the stand up is that it keeps distracting chatter to a daily meeting, so the team can focus on the work for the rest of the day so if it turns into a daily calendar read-out, don't be afraid to change it up and get creative.
These questions highlight progress and help flag team blockers. Also, it strengthens the team when everyone shares the progress they're contributing to the team. The daily reinforcement of sharing individual successes and plans keeps everyone excited about the team's overall contribution to the organization. At the individual level, it's important to walk into the day's stand-up knowing what you're going to say. It keeps the energy of the stand-up high and everyone engaged. Two great filters that can be used together to help prepare for stand-up are "only my issues" and "recently updated." When these two filters are used together, they show the issues assigned to you and that have been updated in the last day.
Make sure each team member gives updates for stories they are working on during stand up calls, if any reason they cannot make it they should update the progress notes for each story. We have four status for each story

- To Do - List of stories agreed and assigned to the team member during sprint planning or during sprint
- In Progress - Stories that team member is spending time to complete tasks as agreed in acceptance criteria
- In Review - Stories which are completed from team member's prospective, however needs peer review/manager review before moving to Done.
- Done - Stories which satisfy all acceptance criteria.


Projects / DevOps and Cloud / DevOps and Cloud Board

DevOps - FY19 Q2 - Sprint 5

1 day remaining   Complete sprint

Create initial DevOps UI that will allow us to configure a provision request and automatically provision tools. Add the foundational tools (ELK and Prometheus) as part of the platform provisioning.

Only My Issues   Recently Updated

TO DO	IN PROGRESS	IN REVIEW	DONE
Dean Chin 3 issues			
Divyanand Bavada 3 issues			
Internal Practice Improvement - Confluence CC Directory Restructure - Review Internal Practice Review 1 DC-134 DB	DevOps Practice - Internal Execution Model - Sprint / Roadmap Define Internal Execution Model 2 DC-14 DB Automate ELK Provisioning and Configuration with Docker container 8 DC-128 DB		
Farid Roshan 3 issues			
Update Jira provision to use SQL instead of H2 DB DP - Tool Provisioning 6 DC-127 FR	Determine best way to collect Jira data and what to collect AI/ML - Collectors 4 DC-123 FR	Update Jira Workflow to include a Resolved State Define Internal Execution Model 1 DC-132 FR	
Vijayakumar Sampath 2 issues			
Automate Prometheus / Grafana Provisioning and Configuration DP - Tool Provisioning 8 DC-104	Udemy - CloudFormation DP - Tool Provisioning 5 DC-129		

Sprint review:

Sprint reviews are not retrospectives. A sprint review is about demonstrating the hard work of the entire team: designers, developers, and the product owner. At the end of the sprint, the team gets together for an informal session to view a demo of, or inspect, the increment. The development team showcases the backlog items that are now 'Done' to stakeholders and teammates for feedback. The product owner can decide whether or not to release the increment, although in most cases the increment is released.
This review meeting is also when the product owner reworks the product backlog based on the current sprint, which can feed into the next sprint planning session.
Step 1: Define 'done'
As a regular user of Jira, there's nothing more satisfying to me than moving a task from 'code review' to 'done.' That swoosh of an agile card represents completed work we set out to accomplish as a team. Crossing the finish line a implement work requires good planning, a clear definition of 'done,' and focused execution. Most of this happens during sprint planning, but to have a successful sprint review and sprint, teams need to do a little more than plan. They need to develop a clear culture of how to deliver work as well as what it means to be 'done.'
"Readiness" and "DoD" are Not Static and can Change as Needed Through Time! A culture of delivery
Effective teams bring clear processes and development culture to each and every work item. Use these questions to assess your process, and make sure it's working optimally for your team:

1. Are stories well-defined by the product owner, designer, and the engineering team before implementation?
2. Does everyone understand and live the team's engineering values and culture?
3. Are there clear definitions and requirements around code review, automated testing, and continuous integration to encourage

sustainable, agile development?
4. After the team completes a story, are there many bugs that surface?
5. In other words, does 'done' really mean 'done?

The team's culture around quality and completion should rise above every user story, engineering work item, and bug. This culture is reflective of how the team approaches and delivers software.

1. Defining 'done' on each work item - A clear definition of 'done' helps teams focus on the end goal for each work item. When the product owner adds work to the team's backlog, defining the acceptance criteria is a key part of his or her process.
2. What does it mean for a user story to be complete? Jira team tracks acceptance criteria and testing notes right in line with the rest of the user story inside of Jira. That way, the entire team has a clear view of success on every issue.
3. What are acceptance criteria and testing notes?• Acceptance criteria: metrics the product owner uses to confirm the story has been implemented to his or her satisfaction.• Testing notes: short, focused guidance from the quality assistance team that enables the development engineer to write better feature code and automated tests.

Having well-defined issues during implementation allows everyone to be successful. With Jira, it's easy to add fields in line. As an administrator, just click the 'admin' button on the issue.
Step 2: celebrate the team
Sprint reviews are a great time to celebrate the team and everyone's accomplishments during an iteration. We typically host them on Friday afternoons, while everyone in the office winds down before the weekend.
Sprint reviews are not synonymous with retrospectives, so make sure to host the sprint review after an iteration, but before your retrospective. External participants are always welcome to join, but the meeting usually consists of the product owner, the full development team, and the scrum master. As a best practice, we recommend spending 30 minutes to an hour for each iteration in the meeting.We love sprint reviews because they protect the health and morale of the team. Sprint reviews are all about team building. The review isn't adversarial, it's not an exam—it's a collaborative event across the team in which people demo their work, field questions, and get feedback.
If a sprint review doesn't become a positive activity across the team, it may be indicative of:

1. The team taking on too much work and not completing it during an iteration
2. The team struggling with existing technical debt.
3. Features not being developed sustainably to ensure new bugs are not introduced into the codebase
4. The team's development practices aren't as tuned as they could be
5. The product owner is changing priorities within an iteration, and the development team is sidelined by scope creep.

Note: every team has a difficult iteration sometimes. Take the time to understand why an iteration changes in the team's retrospective and create a plan to address issues in the next sprint.
Step 3: Reach across geographies –
Companies with distributed teams have special challenges around scaling agile ceremonies across geographies. Sprint reviews are no exception. Seeing a feature demo first-hand by the developer strengthens the team in two ways:

1. Product Understanding: the entire team gets to hear the intention, rationale, and implementation of the feature. It broadens everyone's understanding of the entire product.
2. Team Building: videos create more personal connections across the team. Each of us gets to see who's behind every aspect of a product. The bridges created by this practice makes us a tighter, more cohesive group despite geographies.

This is a good opportunity for the management to see the work completed and participate in the review activities to get an updated insight on the alignment of the work, and what they teams needs in order to succeed.

- Good Sprint Review Practices
- Keep it as an informal meeting, where the "DONE" increment is showcased on machines where the end-users can interact with the newly completed work.
- Solicit and gather feedback (over the completed work) and ideas (for the next increment).
- Let the Development Team communicate directly with end-users and stakeholders and gets direct feedback from them.
- Make sure the Product Backlog is updated after or during the Sprint Review, based on the gathered ideas.
- Try your best to have end-users attend the session (or at least Reps closest to them).

Sprint retrospective:

The retrospective is where the team comes together to document and discuss what worked and what didn't work in a sprint, a project, people or relationships, tools, or even for certain ceremonies. The idea is to create a place where the team can focus on what went well and what needs to be improved for the next time, and less about what went wrong.
Agile teams use retrospectives to reflect, learn, and adapt their way of working. Facilitating retrospectives matters, it's important to have a skilled facilitator to assure that retrospectives become effective. An agile retrospective, or sprint retrospective as Scrum calls it, is a practice used by teams to reflect and become better in what they do. They are a great way to continuously improve the way of working.
Getting feasible actions out of a retrospective and getting them done helps teams to learn and improve to make the retrospective effective, facilitators should focus on the following:

- Establishing an open and honest culture in the meeting
- Ensure that all team members participate in the meeting.
- Assure that the team establishes a shared understanding of how things went.
- Help the team to decide upon the vital few actions that they will take

Here are best seven practices that recommend for facilitating retrospectives to make them valuable for the participants.

1. Many retrospective facilitators use the prime directive to establish a safe culture where team members speak up and will be open and honest. It sets the assumption that team member did the best they could possibly do, and that the purpose of the retrospective is not to blame people.
2. A retrospective facilitator should not have a personal stake in the outcome of the meeting. The team has to decide which actions they will do, the facilitator should not influence their decision. If the Scrum master is facilitating the retrospective this can be a challenge, as (s)he is also a team member.
3. Retrospective facilitators must be able to deal with negative issues. Help the team to focus on the issue and to understand them and don't blame any team members for what has happened. This requires strong communication skills, being able to pay attention to verbal and non-verbal communication. Feelings matter in agile retrospectives.
4. Dealing with silence is another skill that facilitators should have. Silence helps people to think, to reflect, or to accept things that have happened. Sometimes the best thing a facilitator can do is to say and do nothing and give space to the team members to come up with their ideas.
5. A facilitator should focus on the process of the retrospective meeting. They should assure that exercises are done in an appropriate way, keep time, recognize and deal with retrospective smells, and help team members in performing activities that are needed to do the retrospective effectively.
6. A facilitator should serve their team in the meeting, but should not lead people based on their own opinion or ideas on what the team should do. It is important that a facilitator remains independent, which again can be challenging when the facilitator is also the Scrum master of the team.
7. Active listening can be useful in retrospective meetings, for instance when the facilitator recaps what is being said and checks if it has been understood by all attendants. It helps to build a shared understanding in the team.

https://www.scrum.org/resources/scrum-glossary

Sprint Reports

Release Notes:

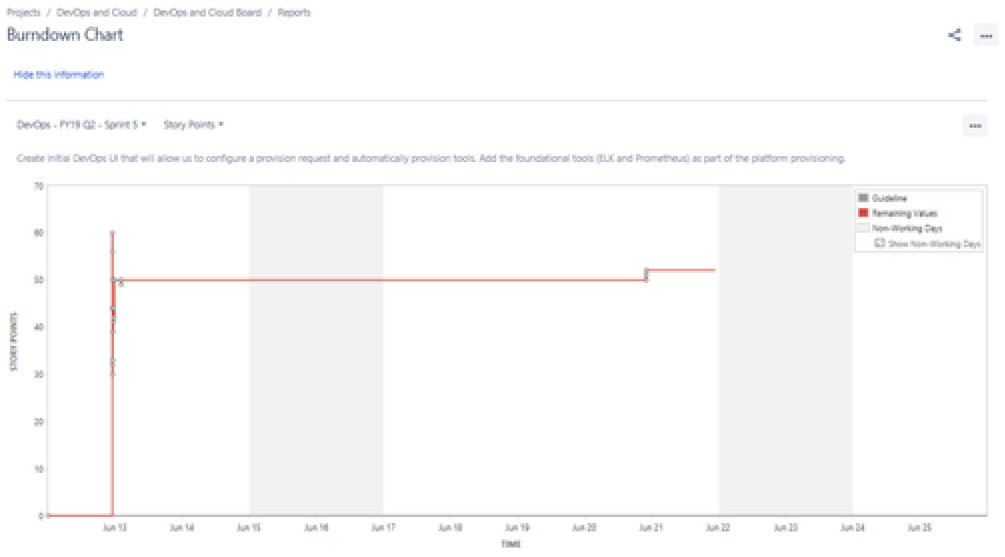
Release notes are a valuable way to communicate the work that has been done.
To assemble our release notes, we would have to comb through all of the Pull Requests and commits made between versions to compile a list of changes. Then, we would need to write a few sentences explaining each change. There are usually multiple developers on a team, and sometimes explaining someone else's code can be difficult.
In Current team structure we are planning to do release notes by monthly. Each month we want to measure the progress and value added for our epics. Each story should have Fix Version selected for the month.
Generating **release notes** in JIRA projects

1. Go to your project and click Project settings.
2. Select Versions in the project sidebar. If you're using Jira Software, select Releases.
3. Select the Version whose release notes you wish to generate by clicking on its name in the list.
4. Click Release Notes.
5. Click Configure Release Notes to select the version and style.
6. Click Create to generate the release notes.

Sample Release Note # Release Notes
New Features* Automate ELK Provisioning and Configuration with Docker container (DC#128)
Updates* Description of the update (DC#218)
Other* Description of the update (#PR_NUMBER)

Burn-down Chart:

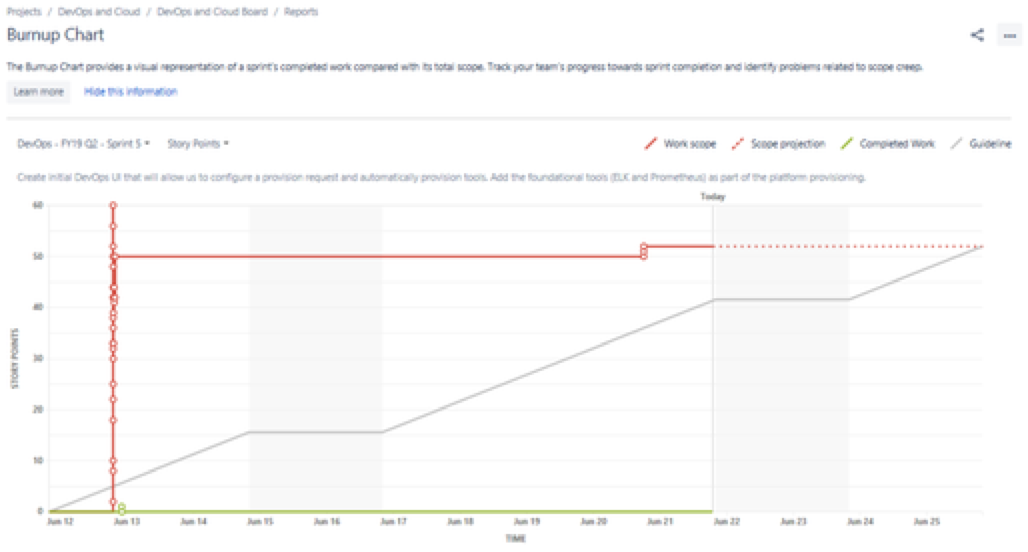
A chart which shows the amount of work which is thought to remain in a backlog. Time is shown on the horizontal axis and work remaining on the vertical axis. As time progresses and items are drawn from the backlog and completed, a plot line showing work remaining may be expected to fall. The amount of work may be assessed in any of several ways such as user story points or task hours. Work remaining in Sprint Backlogs and Product Backlogs may be communicated by means of a burn-down chart.



Create initial DevOps UI that will allow us to configure a provision request and automatically provision tools. Add the foundational tools (ELK and Prometheus) as part of the platform provisioning.

Burn-up Chart:

A chart which shows the amount of work which has been completed. Time is shown on the horizontal axis and work completed on the vertical axis. As time progresses and items are drawn from the backlog and completed, a plot line showing the work done may be expected to rise. The amount of work may be assessed in any of several ways such as user story points or task hours. The amount of work considered to be in-scope may also be plotted as a line; the burn-up can be expected to approach this line as work is complete

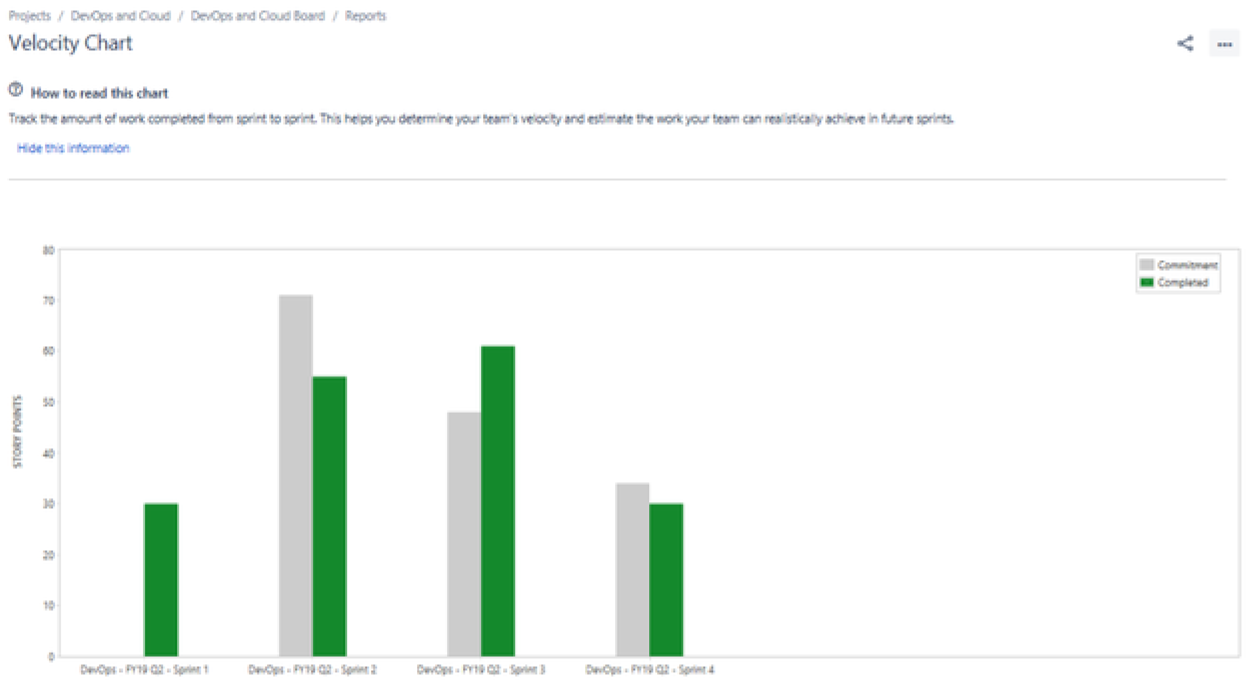


The Burnup Chart provides a visual representation of a sprint's completed work compared with its total scope. Track your team's progress towards sprint completion and identify problems related to scope creep.

Create initial DevOps UI that will allow us to configure a provision request and automatically provision tools. Add the foundational tools (ELK and Prometheus) as part of the platform provisioning.

Velocity Report

Velocity report indication of the average amount of Product Backlog turned into an Increment of product during a Sprint by a Scrum Team, tracked by the Development Team for use within the Scrum Team.



Projects / DevOps and Cloud / DevOps and Cloud Board / Reports

Velocity Chart

How to read this chart

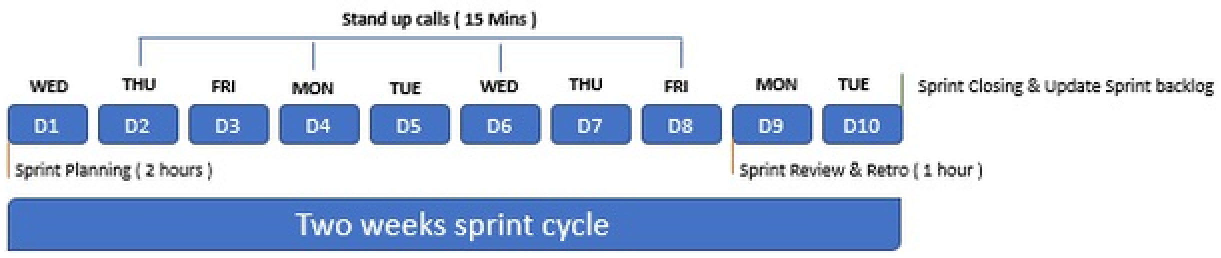
Track the amount of work completed from sprint to sprint. This helps you determine your team's velocity and estimate the work your team can realistically achieve in future sprints.

Hide this information

STORY POINTS
0
10
20
30
40
50
60
70
80
Commitment
Completed
DevOps - FY19 Q2 - Sprint 1
DevOps - FY19 Q2 - Sprint 2
DevOps - FY19 Q2 - Sprint 3
DevOps - FY19 Q2 - Sprint 4

DevOps and Agile team will have two-week timebox sprint.

1. Alternate Wednesday will be start of spring.
2. Will have stand up call twice a week
3. First day of sprint will have 2 hours or Spring Planning meeting and discussion for items selection from product backlog
4. 8th day of sprint will have 1 hours of sprint review/retrospective meeting.



Stand up calls (15 Mins)
WED
THU
FRI
MON
TUE
WED
THU
FRI
MON
TUE
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
Sprint Closing & Update Sprint backlog
Sprint Planning (2 hours)
Sprint Review & Retro (1 hour)
Two weeks sprint cycle